

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«Пензенский государственный университет»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

вычислительной техники

Кафедра

МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Мобильное программное приложение для
поддержки процессов функционирования
ветеринарной клиники**

Студент

(подпись, дата)

Чередник Марина Олеговна

(ФИО полностью)

Руководитель

(подпись, дата)

Князев В.Н.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«20» декабря 2024 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Чередник Марина Олеговна гр. 21ВП2 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Мобильное программное приложение для поддержки процессов функционирования ветеринарной клиники

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Разработать приложение для мобильных устройств, обеспечивающее поддержку процессов функционирования ветеринарной клиники.

2) Среда выполнения приложения: операционная система Android 10 и выше

Среда выполнения сервера: операционная система Windows 10 и выше

Языковые средства разработки: C#, SQL, XAML, UML, Linq

Инструментальные средства разработки: Microsoft Visual Studio 2022,

Microsoft SQL Server 2019, Microsoft Project, Visual Paradigm

Технологии разработки: RUP, ООП, MAUI, Entity Framework

3) Основные функции мобильного приложения:

а) запись на прием к ветеринарному врачу;

б) просмотр истории приемов у ветеринарного врача;

в) просмотр записей на прием клиентов и медицинских карточек питомцев;

г) просмотр доступных услуг и их цен;

д) просмотр рекомендаций ветеринарного врача;

е) добавление расписания ветеринарного врача;

ж) изменение карточки питомца

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 188 листов, 50 рисунков, 32 таблицы, 25 использованных источника, 7 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ANDROID, C#, MAUI, .NET, SQL, LINQ, UML, RUP, MS SQL SERVER, ВЕТЕРИНАРНАЯ КЛИНИКА, ЗАПИСЬ НА ПРИЕМ

Объектом разработки является мобильное приложение для поддержки процессов функционирования ветеринарной клиники.

Цель работы – разработка мобильного приложения, позволяющего клиентам записываться на прием к ветеринарному врачу, а ветеринарным врачам вести медицинские карты питомцев и управлять своим расписанием.

Разработка серверной части приложения производилась на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2022. В качестве СУБД была использована СУБД MS SQL Server 2019.

Разработка мобильного приложения производилась на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2022.

Разработка проведена с использованием операционной системы Windows 10. Приложение предназначено для использования под управлением ОС Android версии 10.0 и выше, а его серверная часть работает под управлением операционной системы Windows 10 и выше.

					ПГУ 09.03.04 – 08БР212.28 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Мобильное программное приложение для поддержки процессов функционирования ветеринарной клиники. Пояснительная записка	Лит.	Лист	Листов
Разраб.	Чередник М.О.						4	188
Пров.	Князев В.Н.					Группа 21ВП2		
Реценз.								
Н. контр.	Такташкин Д.В.							
Утв.	Козлов А.Ю.							

Содержание

Введение.....	7
1 Анализ предметной области и требований	9
1.1 Анализ предметной области и постановка задачи	9
1.2 Анализ программных продуктов аналогичного назначения.....	9
1.2.1 Web-приложение «Доктор Зоо».....	10
1.2.2 Web-приложение «Ветеринарная клиника врачей Сумбаевых»...	11
1.2.3 Web-приложение «Счастливый слон»	12
1.2.4 Сравнение аналогов	12
1.3 Анализ целевой аудитории	13
1.4 Анализ функциональных требований.....	16
1.5 Анализ нефункциональных требований.....	21
1.6 Планирование разработки и оценка бюджета.....	22
2 Проектирование программного обеспечения	34
2.1 Выбор технологий и инструментальных средств разработки.....	34
2.2 Проектирование архитектуры программного обеспечения	35
2.3 Проектирование базы данных	36
2.3.1 Концептуальное проектирование базы данных	36
2.3.2 Логическое проектирование базы данных.....	38
2.4 Проектирование серверной части приложения	40
2.5 Проектирование интерфейса пользователя.....	41
3 Реализация программного обеспечения	57
3.1 Реализация базы данных	57
3.2 Реализация серверной части приложения	61
3.3 Реализация мобильного приложения.....	64
3.4 Тестирование программного обеспечения	67
3.4.1 Тестирование серверной части приложения	67
3.4.2 Тестирование клиентской части приложения	70
3.4.3 Метрики кода.....	71
Заключение	77

Список использованных источников	79
Приложение А (справочное) Глоссарий	82
Приложение Б (обязательное) Исходный код скрипта создания базы данных	84
Приложение В (обязательное) Исходный код серверной части приложения	88
Приложение Г (обязательное) Исходный код мобильного приложения	126
Приложение Д (рекомендуемое) Результаты тестирования серверной части приложения	168
Приложение Е (рекомендуемое) Исходный код функций тестирования.....	178
Приложение Ж (рекомендуемое) Результаты тестирования мобильного приложения	181

Введение

В современном мире информационные технологии стремительно развиваются и находят широкое применение во всех сферах человеческой деятельности, включая здравоохранение, образование, бизнес и сферу обслуживания. Особенно заметен рост использования мобильных приложений, которые позволяют значительно повысить удобство, скорость и качество предоставляемых услуг.

Россия также активно включилась в процесс цифровой трансформации различных сфер деятельности. В связи с тем, что интеграция информационных технологий способна многократно повысить удобство и эффективность этих самых процессов, правительство Российской Федерации в 2017 году утвердило программу «Цифровая экономика Российской Федерации». Данная программа подразумевает «развитие цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост и экономический суверенитет» [1].

Современные тенденции цифровизации демонстрируют стремительный рост использования мобильных технологий как в России, так и во всем мире. Широкое внедрение смартфонов и планшетов с постоянным доступом в интернет изменило способ взаимодействия пользователей с цифровыми сервисами. Поскольку мобильный интернет значительно удобнее, традиционный доступ к сети с помощью стационарных компьютеров постепенно оттесняется. Статистические данные, приведенные в программе "Цифровая экономика", наглядно иллюстрируют этот процесс: еще в 2016 году показатель использования мобильных телефонов достиг 159,95 устройства на 100 жителей, при этом 71,29% населения уже активно пользовались мобильным интернетом [2]. И эта тенденция со временем только усиливается.

На фоне этих тенденций растёт спрос на специализированные мобильные решения, упрощающие взаимодействие между организациями и клиентами.

Ветеринарная медицина – одна из таких сфер, где автоматизация и цифровизация процессов позволяют улучшить качество обслуживания, упростить запись на приём, обеспечить доступ к медицинской информации и повысить лояльность клиентов.

Опираясь на приведённые выше факторы можно сделать вывод о том, что в современном мире мобильные приложения необходимы в любой отрасли, в том числе и ветеринарной, и компании, не имеющие мобильного приложения, проигрывают в удобстве обслуживания клиентов компаниям, которые его имеют, а значит недополучают прибыль.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Интерфейс разрабатываемого мобильного приложения должен быть интуитивно понятным и максимально удобным для пользователя. Приложение позволит клиентам ветеринарной клиники изучить услуги, которые предоставляет клиника, оценить их стоимость, а также легко и быстро записаться на прием к ветеринарному врачу. Кроме того, после приема владельцы питомцев смогут просмотреть в приложении рекомендации ветеринарного врача.

Таким образом, в приложении должны быть представлены следующие основные функции:

- регистрация новых пользователей;
- вход зарегистрированных пользователей в систему;
- запись на прием;
- просмотр доступных услуг и их цен;
- просмотр рекомендаций ветеринарного врача.

Помимо рядовых пользователей в системе будут присутствовать ветеринарные врачи и администраторы, которые будут иметь более широкие возможности действий. Ветеринары смогут создавать свое расписание для записи клиентов, а также вести медицинские карты питомцев и загружать рекомендации после приема. Администраторы будут дополнительно иметь доступ к редактированию услуг и цен и управлению персоналом клиники.

1.2 Анализ программных продуктов аналогичного назначения

В настоящее время практически у каждой ветеринарной клиники существует сайт, предоставляющий информацию об услугах клиники, однако очень мало клиник используют мобильные приложения. Рассмотрим примеры приложений ветеринарных клиник и произведём для них сравнительный анализ.

1.2.1 Web-приложение «Доктор Зоо»

Доктор Зоо сеть ветеринарных клиник в Пензе и Пензенской области [3]. Мобильного приложения у клиники нет, но клиника обладает своим собственным сайтом, который предоставляет информацию о широком спектре предлагаемых ветеринарных услуг: приём, диагностику, лечение, вызов ветеринарного врача на дом, груминг и продажу продукции собственного производства. Личного кабинета на сайте нет. С помощью панели навигации можно переходить по вкладкам и изучить информацию об услугах, о клинике, сотрудниках, новостях клиники, а также найти контакты клиники для связи. Запись в клинику осуществляется по телефону, возможности записи на прием на сайте нет.

Главная страница web-приложения «Доктор Зоо» представлена на рисунке 1.

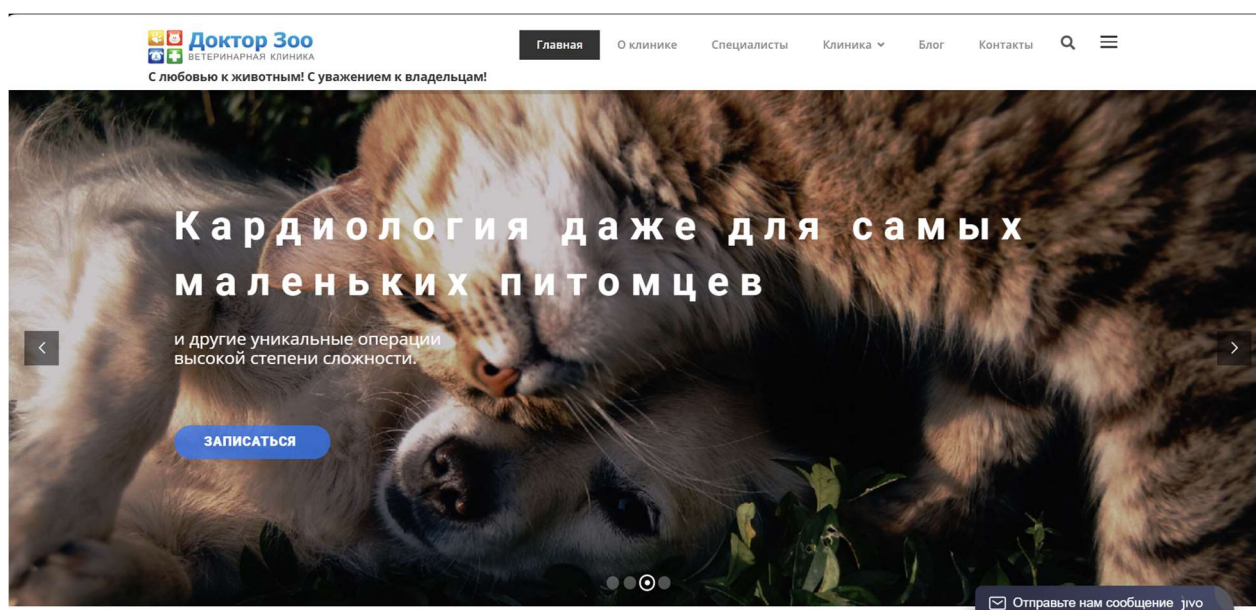


Рисунок 1 – Главная страница web-приложения «Доктор Зоо»

К достоинствам приложения относится возможность ознакомиться со всей необходимой для клиентов информацией.

К недостаткам приложения относится отсутствие возможности авторизации, просмотра рекомендаций ветеринарного врача и записи на прием.

1.2.2 Web-приложение «Ветеринарная клиника врачей Сумбаевых»

Ветеринарная клиника врачей Сумбаевых – это частная клиника в Пензе, предоставляющая широкий спектр ветеринарных услуг [4]. Web-приложение позволяет ознакомиться с информацией о клинике, сотрудниках, услугах и ценами на них. Сайт не поддерживает авторизацию, но предоставляет функционал записи на первичный и повторный приемы. С помощью удобной формы можно выбрать врача, специализацию или услугу, а затем ознакомиться с доступным временем для записи и осуществить запись на прием.

Главная страница web-приложения «Ветеринарная клиника врачей Сумбаевых» представлена на рисунке 2.

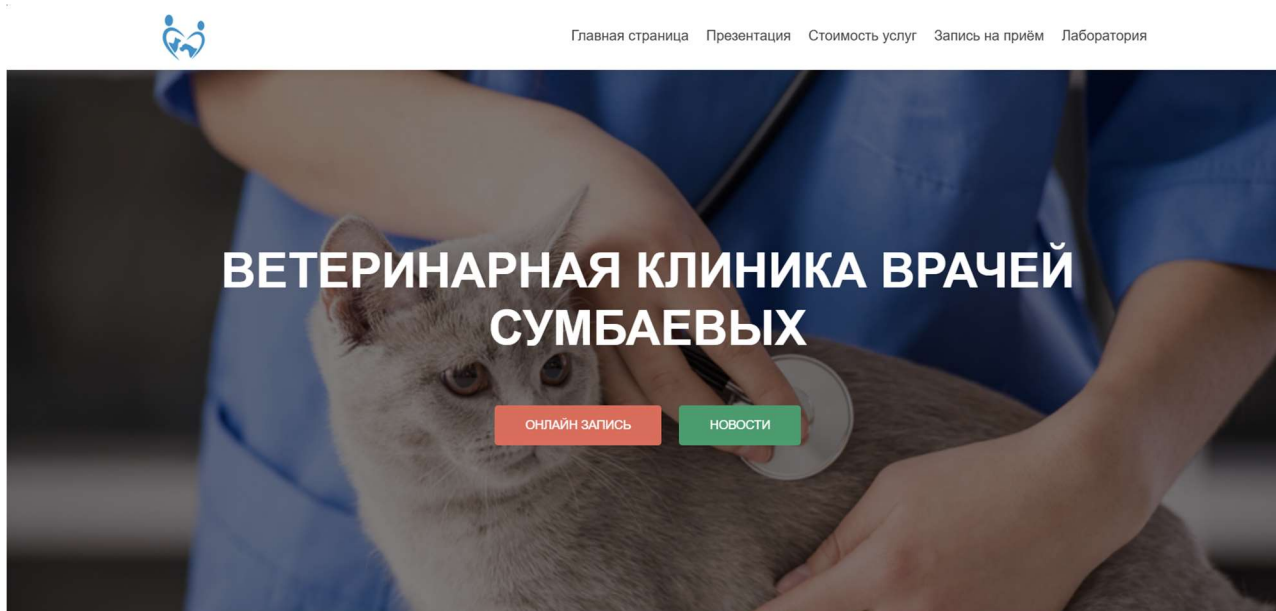


Рисунок 2 – Главная страница web-приложения «Ветеринарная клиника врачей Сумбаевых»

К достоинствам приложения относится возможность онлайн-записи на прием к ветеринарному врачу, а также возможность ознакомиться со всей необходимой информацией, в том числе список услуг и прайсом.

К недостаткам приложения относится отсутствие возможности авторизации и просмотра рекомендаций ветеринарного врача.

1.2.3 Web-приложение «Счастливый слон»

Счастливый слон – зооцентр, находящийся в Пензе и Саранске [5]. Web-приложение позволяет ознакомиться не только с услугами, но и товарами их магазина. Также сайт предоставляет информацию о сотрудниках, новостях и акциях, контактах зооцентра. Есть возможность ознакомиться с отзывами о зооцентре. Сайт, как и предыдущие аналоги, не поддерживает авторизацию. Функционала записи на прием через сайт тоже нет, запись осуществляется по телефону. Главная страница web-приложения «Счастливый слон» представлена на рисунке 3.

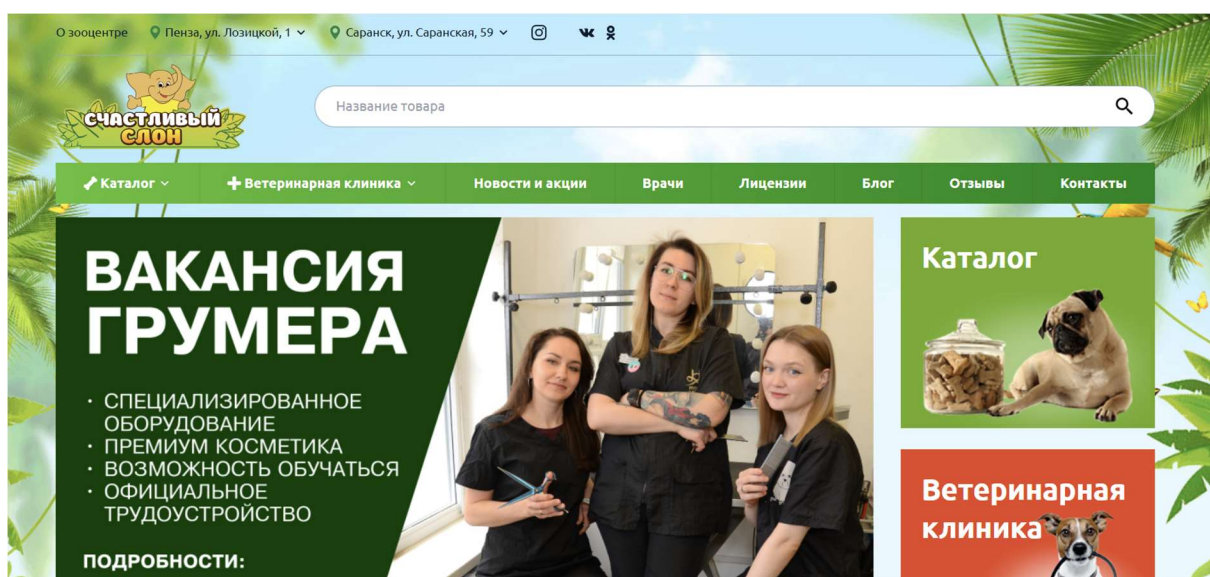


Рисунок 3 – Главная страница web-приложения «Счастливый слон»

К достоинствам приложения относится возможность ознакомиться со всей необходимой информацией, а также удобный поиск по товарам.

К недостаткам приложения относится отсутствие возможности авторизации и онлайн-записи на прием.

1.2.4 Сравнение аналогов

Сравнительный анализ рассмотренных выше приложений представлен в таблице 1.

Таблица 1 – Сравнение аналогов

Приложение	Запись на прием к ветеринарному врачу	Вход в личный кабинет	Просмотр истории посещений и рекомендаций ветеринарного врача	Информация об услугах и ценах ветеринарной клиники	Наличие мобильного приложения
Доктор Зоо	Нет	Нет	Нет	Да	Нет
Ветеринарная клиника врачей Сумбаевых	Да	Нет	Нет	Да	Нет
Счастливый слон	Нет	Нет	Нет	Да	Нет
Разрабатываемое приложение	Да	Да	Да	Да	Да

Таким образом, в результате анализа систем аналогичного назначения было принято решение о разработке собственного приложения, которое будет лишено недостатков, выявленных у рассмотренных выше приложений.

1.3 Анализ целевой аудитории

Целевой аудиторией проекта являются владельцы домашних животных и работники ветеринарных клиник. По данным Центра питания и благополучия животных на 2023 год, домашних животных в России имеют 56% семей, а хозяевами домашних животных являются 68 млн россиян старше 14 лет. Также в России растет количество домохозяйств (на 2023 год 66,9 млн, или на 2,5% по отношению к 2020 году), и это способствует увеличению численности домашних питомцев. Популяция домашних животных в России достигла 74,8 млн, а рост численности домашних питомцев на 2023 год к 2020 году составил 11% [6].

На рисунке 4 представлена динамика исследования количества домашних животных (кошек и собак) в России в 2017 – 2023 годах.

Динамика исследования за 2017-2020-2023 годы



Рисунок 4 – Количество кошек и собак в России

На рисунке 5 представлены доли основных операционных систем смартфонов на мировом рынке за последние 2 года [7].

Global Smartphone Sales Share by Operating System

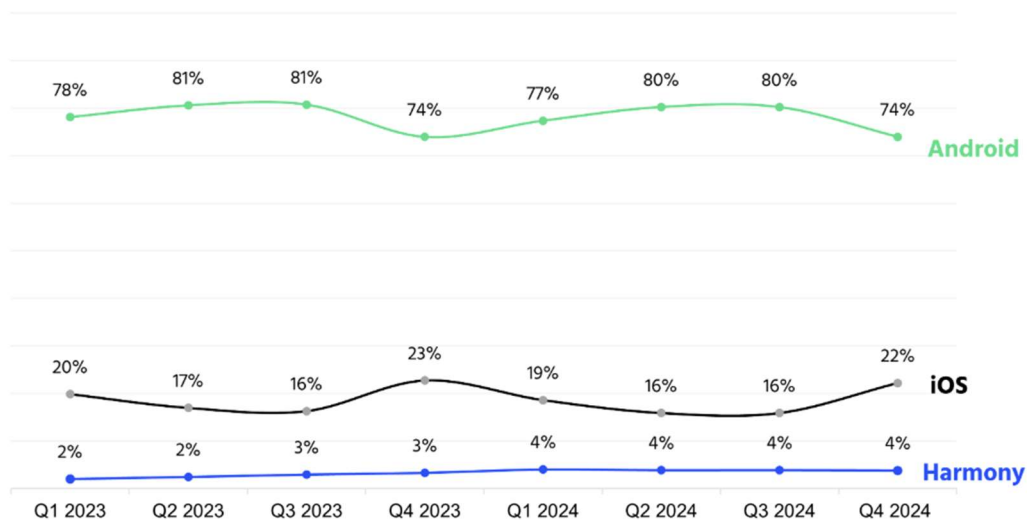


Рисунок 5 – Доли операционных систем смартфонов на мировом рынке за последние 2 года

Из графика можно увидеть, что подавляющее большинство смартфонов на мировом рынке используют операционную систему Android. Пользователи

Android составляют 74% от общего количества пользователей [8]. Несмотря на рост использования операционной системы iOS компании Apple, показатели данной операционной системы значительно ниже, чем у операционной системы Android, и составляют 22%. В России доля смартфонов Apple ниже, чем в среднем по планете. Поэтому было принято решение разрабатывать приложение для операционной системы Android. Так же в России на данный момент развиваются и другие операционные системы отечественной разработки, такие как Аврора. Аврора – российская мобильная операционная система на базе Linux. Она создана для корпоративных смартфонов и планшетов. Система разработана для внедрения на предприятиях, но также может использоваться и на отдельных устройствах пользователей [9].

Статистика распространения разных версий системы Android за последний год представлена на рисунке 6 [10].

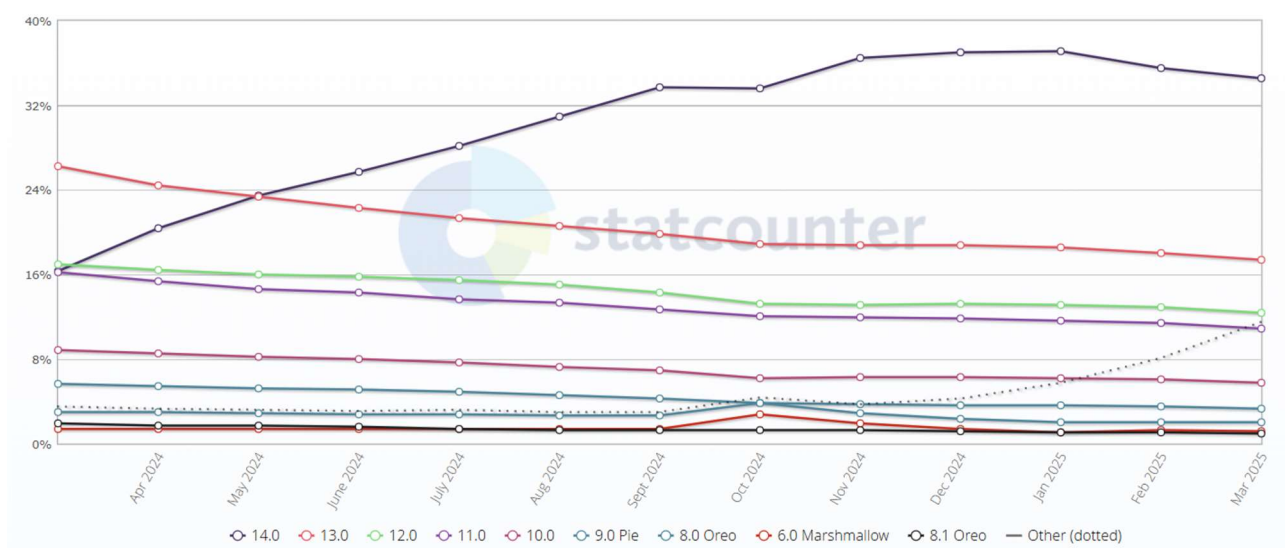


Рисунок 6 – Распространение различных версий системы Android

Из приведённой диаграммы можно увидеть, что более 90 % пользователей системы Android использует версии не старше 10.0. Поэтому приложение будет разрабатываться таким образом, чтобы оно успешно функционировало для Android 10.0 и более новых версий системы.

1.4 Анализ функциональных требований

Приложение может применяться владельцами домашних животных и работниками ветеринарной клиники. Приложение позволяет создавать и отменять записи на прием к ветеринарному врачу, просматривать рекомендации ветеринарного врача после приема и историю посещений. Также клиент клиники может ознакомиться с услугами, предоставляемыми клиникой, и ценами на них.

Пользователи могут зарегистрироваться в приложении, после чего смогут авторизоваться в системе.

После входа в личный аккаунт клиент может добавить своих питомцев и редактировать информацию о них. После добавления питомца клиент может записать его на прием к ветеринарному врачу.

Для записи к ветеринарному врачу клиент должен выбрать услугу. После выбора услуги будет доступен выбор врача, предоставляющего данную услугу. Затем клиент должен выбрать удобную для него дату и время из доступных. Также клиенту нужно выбрать питомца, которого он записывает на прием.

Доступное время и даты для записи составляются ветеринарным врачом. Для этого врач должен указать свое расписание в приложении. Врач также может посмотреть записи к нему, добавлять рекомендации после приема, доступные клиенту для просмотра, а также добавлять медицинскую карточку питомца. Медицинские карточки привязаны к посещению, ветеринарный врач может указать цель визита, вес и температуру животного, жалобы и симптомы, диагноз, рекомендации и статус приема. Кроме того, врачу доступна возможность посмотреть историю посещений питомца.

Администратор клиники имеет тот же функционал, что и ветеринарный врач. Дополнительно администратор может изменять услуги и цены, добавлять и удалять их, а также добавлять и удалять ветеринарных врачей.

В результате анализа функциональных требований к разрабатываемому приложению была создана диаграмма вариантов использования, представленная на рисунке 7.

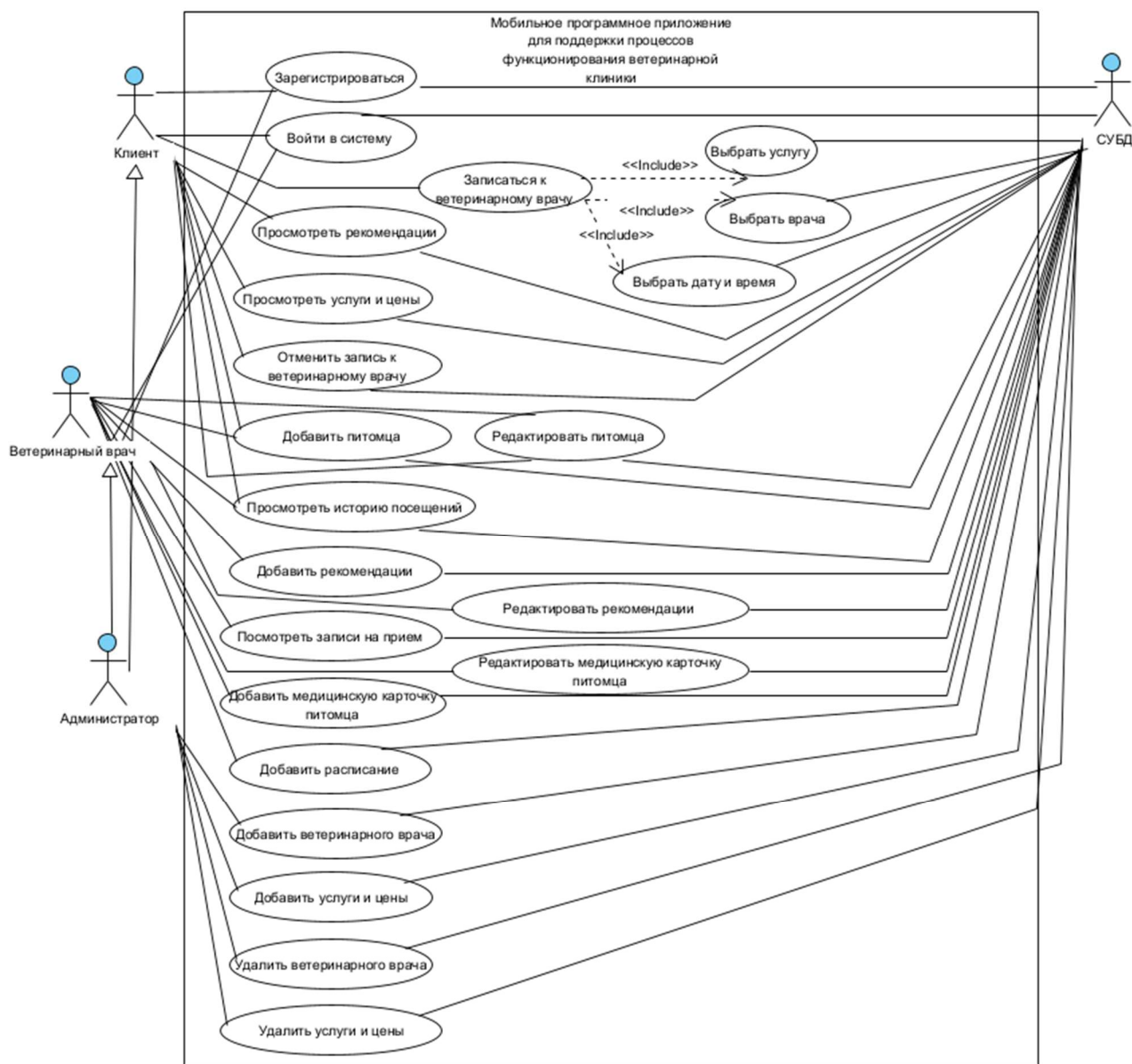


Рисунок 7 – Диаграмма вариантов использования мобильного приложения

На основе получившейся диаграммы составим спецификации для основных вариантов использования: «Записаться к ветеринарному врачу», «Зарегистрироваться», «Войти в систему», «Добавить питомца» и «Добавить расписание».

Спецификация для прецедента «Записаться к ветеринарному врачу» представлена на рисунке 8.

Наименование: записаться к ветеринарному врачу
ID: 1
Краткое описание: клиент записывает питомца на прием к ветеринарному врачу
Действующие лица: клиент, СУБД
Предусловие: клиент зарегистрировался в системе и добавил питомца
Основной поток: <ol style="list-style-type: none"> 1. Клиент выбирает услугу 2. Система ищет информацию о врачах, оказывающих выбранную услугу 3. Система отображает найденную информацию о врачах 4. Клиент выбирает ветеринарного врача 5. Система ищет информацию о доступных датах для записи к ветеринарному врачу 6. Система отображает найденную информацию о датах для записи 7. Клиент выбирает дату для записи 8. Система ищет информацию о доступном времени для записи к врачу в выбранную дату 9. Система отображает найденную информацию о времени для записи 10. Клиент выбирает время для записи 11. Клиент выбирает питомца 12. Система проверяет корректность введенных данных 13. ЕСЛИ время доступно и питомец выбран <ol style="list-style-type: none"> 13.1. Система записывает клиента на прием, сообщает об этом клиенту 14. ИНАЧЕ <ol style="list-style-type: none"> 14.1. Система сообщает, что запись на прием не удалась
Постусловие: клиент записал питомца на прием к ветеринарному врачу

Рисунок 8 – Спецификация варианта использования «Записаться к ветеринару»

Спецификация для прецедента «Зарегистрироваться» представлена на рисунке 9.

Наименование: зарегистрироваться
ID: 2
Краткое описание: пользователь регистрирует новый аккаунт
Действующие лица: клиент/ветеринарный врач/администратор, СУБД
Предусловие: приложение запущено
Основной поток: <ol style="list-style-type: none"> 1. Пользователь выбирает регистрацию 2. Пользователь вводит логин, пароль, ФИО, email, номер телефона. 3. Система проверяет логин и пароль на корректность 4. ЕСЛИ логин и пароль корректны <ol style="list-style-type: none"> 4.1. Система проверяет наличие аккаунта с введенным логином 4.2. Система сообщает о наличии аккаунта с введенным логином 4.3. ЕСЛИ такого аккаунта нет <ol style="list-style-type: none"> 4.3.1. СУБД добавляет в БД нового пользователя 4.3.2. Система сообщает об успешной регистрации
Постусловие: пользователь зарегистрировался

Рисунок 9 – Спецификация варианта использования «Зарегистрироваться»

Спецификация для прецедента «Войти в систему» представлена на рисунке 10.

Наименование: войти в систему
ID: 3
Краткое описание: пользователь входит в систему
Действующие лица: клиент/ветеринарный врач/администратор, СУБД
Предусловие: приложение запущено
Основной поток:
1. Пользователь выбирает авторизацию
2. Пользователь вводит логин и пароль
3. Система проверяет сведения о наличии аккаунта с введенным логином и паролем
4. ЕСЛИ такой аккаунт существует
4.1. Система осуществляет вход пользователя в систему
4.2. Иначе система выводит сообщение о неверном логине или пароле
Постусловие: пользователь вошел в систему

Рисунок 10 – Спецификация варианта использования «Войти в систему»

Спецификация для прецедента «Добавить питомца» представлена на рисунке 11.

Наименование: добавить питомца
ID: 4
Краткое описание: клиент добавляет нового питомца
Действующие лица: клиент/ветеринарный врач/администратор, СУБД
Предусловие: пользователь вошел в систему
Основной поток:
1. Пользователь выбирает добавление питомца
2. Пользователь вводит тип животного, кличку, дату рождения, породу и пол
3. Система проверяет корректность данных
4. ЕСЛИ данные корректны
4.1.1. СУБД добавляет в БД нового питомца
4.1.2. Иначе система сообщает об ошибке
Постусловие: пользователь добавил питомца

Рисунок 11 – Спецификация варианта использования «Добавить питомца»

Спецификация для прецедентов «Добавить расписание» представлена на рисунке 10.

Наименование: добавить расписание
ID: 5
Краткое описание: пользователь входит в систему
Действующие лица: ветеринарный врач, СУБД
Предусловие: приложение запущено
Основной поток:
5. Врач выбирает добавление расписания
6. Врач выбирает дни недели работы
7. Врач выбирает время работы
8. Врач указывает длительность приема
9. Система проверяет данные на корректность
10. ЕСЛИ данные корректны
10.1. Система добавляет слоты для записи в базу данных
10.2. Иначе система выводит сообщение об ошибке
Постусловие: ветеринарный врач добавил расписание

Рисунок 12 – Спецификация варианта использования «Добавить расписание»

Для варианта использования «Войти в систему» составлена диаграмма деятельности, представленная на рисунке 13.

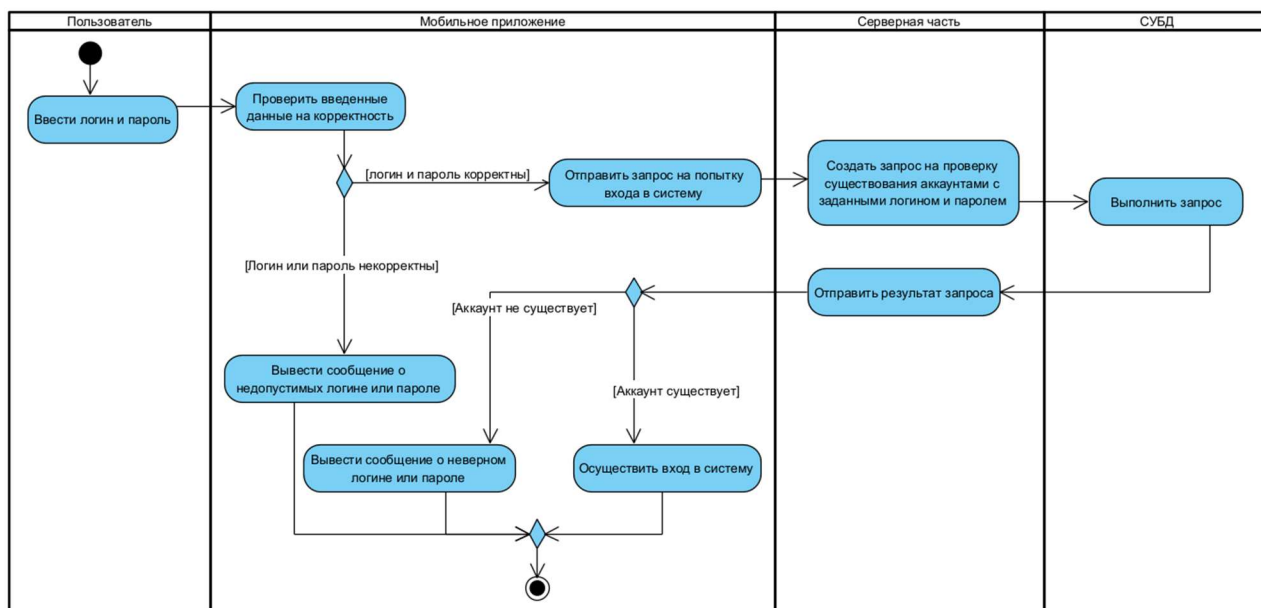


Рисунок 13 – Диаграмма деятельности варианта использования «Войти в систему»

Для варианта использования «Добавить питомца» была составлена диаграмма деятельности, представленная на рисунке 14.

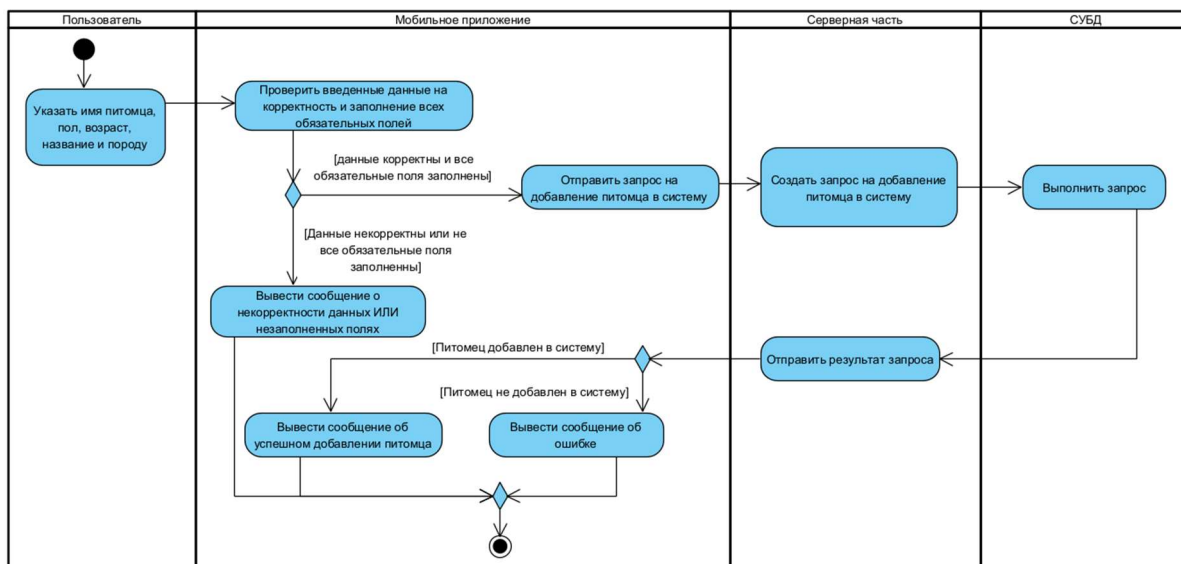


Рисунок 14 – Диаграмма деятельности варианта использования «Добавить питомца»

Для варианта использования «Записаться к ветеринарному врачу» была составлена диаграмма последовательности, представленная на рисунке 15.

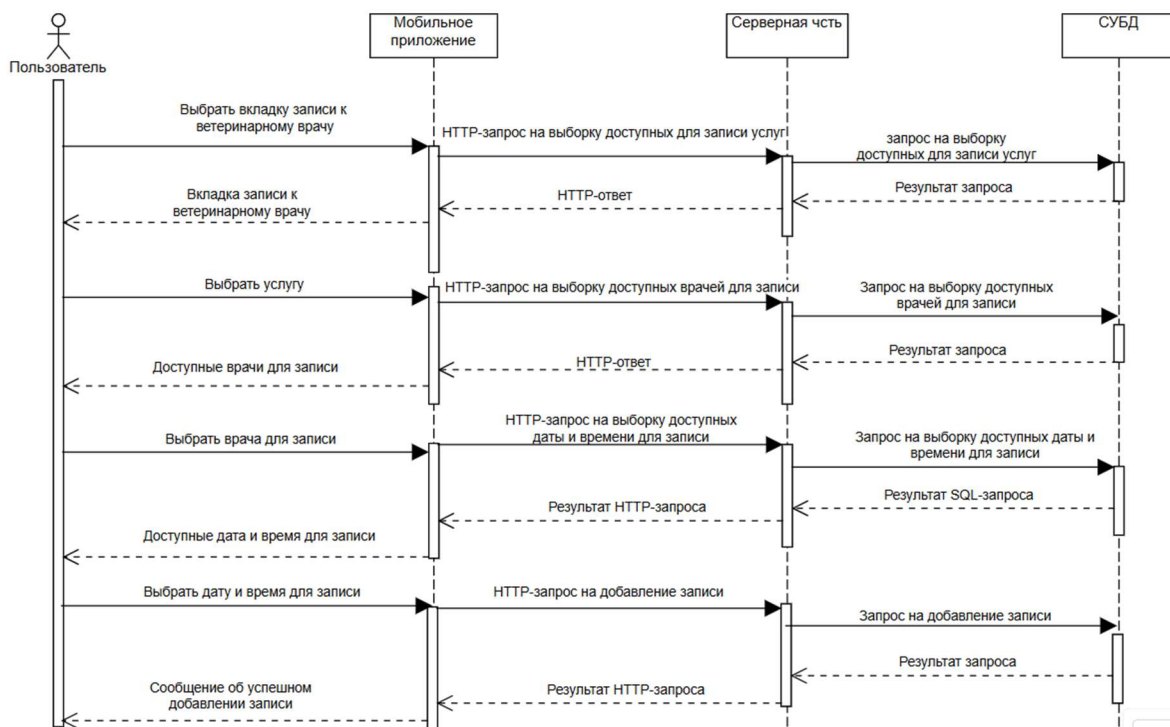


Рисунок 15 – Диаграмма последовательности варианта использования «Записаться к ветеринарному врачу»

В результате анализа функциональных требований были определены основные функции приложения, а также составлены диаграммы вариантов использования, деятельности и последовательности.

1.5 Анализ нефункциональных требований

На данном этапе были определены нефункциональные требования к разрабатываемому программному обеспечению. При разработке должна быть использована такая технология разработки, как RUP [11]. Также при разработке должны применяться принципы ООП.

Разработка серверной части приложения должна быть выполнена на языке программирования C# на платформе ASP.NET в среде программирования Microsoft Visual Studio 2022 [12]. Данные на сервере должны храниться в базе данных, реализованной с использованием СУБД Microsoft SQL Server. Запросы

к базе данных будут осуществляться на языке SQL. Серверная часть должна работать под управлением операционной системы Windows 10 и выше.

Клиентская часть приложения будет реализована на языке программирования C# с использованием кроссплатформенного фреймворка .NET MAUI в среде программирования Microsoft Visual Studio 2022. Она должна представлять собой мобильное приложение с удобным и интуитивно понятным для пользователей интерфейсом. Клиентское приложение должно быть отлажено и должно успешно работать под управлением ОС Android 10.0 и выше.

1.6 Планирование разработки и оценка бюджета

Для качественного выполнения проекта в установленные сроки необходимо правильно распланировать разработку. Это позволит обеспечить выполнение работ над проектом таким образом, чтобы добиться выполнения всех поставленных целей за минимально возможное время, с минимально возможными затратами и с приемлемым качеством [13–14].

Задачи, представляющие собой этапы работы над проектом, представлены в таблице 2.

Таблица 2 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	–	–
2	Анализ требований	–	–
3	Анализ предметной области и постановка задачи	1 день	1
4	Анализ программных продуктов аналогичного назначения	3 дня	1
5	Анализ функциональных требований	2 дня	3; 4
6	Анализ требований к интерфейсу	2 дня	5
7	Анализ целевой аудитории	1 день	6
8	Анализ требований завершён	–	7
9	Выбор технологий разработки ПО	–	–
10	Анализ существующих технологий разработки ПО	2 дня	1
11	Выбор технологий разработки ПО	1 день	10
12	Выбор средств разработки ПО завершён	–	11
13	Проектирование ПО	–	–
14	Проектирование архитектуры приложения	4 дня	8; 12
15	Проектирование БД	2 дня	14

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
16	Проектирование серверной части приложения	2 дня	15
17	Проектирование интерфейса пользователя	3 дня	8; 12
18	Проектирование приложения завершено	–	17; 16
19	Реализация ПО	–	–
20	Реализация БД	7 дней	14; 15
21	Реализация серверной части приложения	20 день	18; 20
22	Реализация мобильного приложения	20 дней	18; 20
23	Реализация завершена	–	21; 22
24	Тестирование	–	–
25	Функциональное тестирование	3 дня	23
26	Модульное тестирование	3 дня	23
27	Тестирование завершено	–	25; 26
28	Составление и оформление документации	7 дней	27
29	Конец проекта	–	28

Таким образом, для реализации проекта нужно выполнить 29 задач.

Ниже в таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Перечень ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист 1	Трудовой	450 руб/ч	600 руб/ч
Программист 2	Трудовой	400 руб/ч	550 руб/ч
Аналитик	Трудовой	400 руб/ч	550 руб/ч
Дизайнер	Трудовой	350 руб/ч	400 руб/ч
Тестировщик	Трудовой	350 руб/ч	400 руб/ч
Компьютер	Материальный	2000 руб	-
Мобильный телефон	Материальный	500 руб	-
Интернет	Затраты	200 руб	-

В таблице 4 приведено распределение ресурсов по задачам.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта	-	-
2	Анализ требований	-	-
3	Анализ предметной области и постановка задачи	Аналитик Компьютер Интернет	40% 3 200 руб
4	Анализ программных продуктов аналогичного назначения	Аналитик Мобильный телефон Интернет	50% 1 шт
5	Анализ функциональных требований	Программист 1 Программист 2 Аналитик	100% 100% 50%
6	Анализ требований к интерфейсу	Дизайнер Аналитик	100% 30%
7	Анализ целевой аудитории	Аналитик	50%

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
8	Анализ требований завершен	-	-
9	Выбор средств разработки ПО	-	-
10	Анализ существующих технологий разработки ПО	Программист 1 Интернет	30%
11	Выбор технологий разработки ПО	Программист 1 Программист 2	50% 50%
12	Выбор средств разработки ПО завершен	-	-
13	Проектирование ПО	-	-
14	Проектирование архитектуры приложения	Программист 1 Программист 2 Аналитик	100% 100% 50%
15	Проектирование БД	Программист 2 Аналитик	70% 50%
16	Проектирование серверной части приложения	Программист 1 Аналитик	100% 30%
17	Проектирование интерфейса пользователя	Программист 1 Дизайнер Аналитик	30% 60% 50%
18	Проектирование ПО завершено	-	-
19	Реализация ПО	-	-
20	Реализация БД	Программист 2	100%
21	Реализация серверной части	Программист 1	100%
22	Реализация мобильного приложения	Программист 1 Программист 2 Мобильный телефон	50% 50% 1 шт
23	Реализация завершена	-	-
24	Тестирование	-	-
25	Функциональное тестирование	Тестировщик Мобильный телефон	70% 1 шт
26	Модульное тестирование	Тестировщик Программист 2	70% 50%
27	Тестирование завершено	-	-
28	Составление и оформление документации	Аналитик Программист 1 Программист 2	100% 50% 50%
29	Конец проекта	-	-

Таким образом, реализацией проекта будут заниматься программист 1, программист 2, аналитик, дизайнер и тестировщик, которые в процессе работы используют три компьютера и доступ в интернет.

Далее необходимо рассчитать длительность реализации проекта.

Для того, чтобы вычислить длительности реализации проекта и его критический путь, был построен сетевой график, который представлен на рисунке 16.

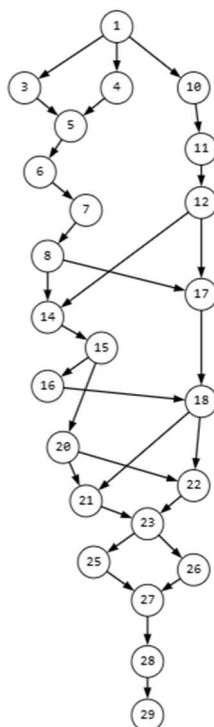


Рисунок 16 – Сетевой график проекта

Теперь необходимо рассчитать критический путь проекта. Для этого необходимо высчитать раннее и позднее времена начала работ для всех задач проекта, и на основе данных сведений определить резерв времени для каждой из задач. В таблице 5 приведены расчеты раннего времени начала работ.

Таблица 5 – Расчёты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	—	—	—	—	0
3	1	0	0	0+0	0
4	1	0	0	0+0	0
5	3	0	1	Max(0+1;0+3)	3
	4	0	3		
6	5	3	2	3+2	5
7	6	5	2	5+2	7
8	7	7	1	7+1	8
10	1	0	0	0+0	0
11	10	0	2	0+2	2

Продолжение таблицы 5

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
12	11	2	1	2+1	3
14	8	8	0	Max(8+0;3+0)	8
	12	3	0		
15	14	8	4	8+4	12
16	15	12	2	12+2	14
17	8	8	0	Max(8+0;3+0)	8
	12	3	0		
18	17	8	3	Max(8+3;14+2)	16
	16	14	2		
20	14	8	4	Max(8+4;12+2)	14
	15	12	2		
21	18	16	0	Max(16+0;14+7)	21
	20	14	7		
22	18	16	0	Max(16+0;14+7)	21
	20	14	7		
23	21	21	20	Max(21+20;21+20)	41
	22	21	20		
25	23	41	0	41+0	41
26	23	41	0	41+0	41
27	25	41	3	Max(41+3;41+3)	44
	26	41	3		
28	27	44	7	44+7	51
29	28	51	0	51+0	51

В таблице 6 рассчитано позднее время начала работ задач.

Таблица 6 – Расчёты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
29	—	—	—	—	51
28	29	51	0	51-0	51
27	28	51	7	51-7	44
26	27	44	3	44-3	41
25	27	44	3	44-3	41
23	25	41	3	Min(41-3; 41-3)	38
	26	41	3		
22	23	38	0	38-0	38
21	23	38	0	38-0	38
20	21	38	20	Min(38-20;38-20)	18
	22	38	20		
18	21	38	20	Min(38-20;38-20)	18
	22	38	20		
17	18	18	3	18-3	15
16	18	18	2	18-2	16

Продолжение таблицы 6

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
15	16 20	16 18	2 7	Min(16-2;18-7)	11
14	15 20	11 18	2 7	Min(11-2;18-7)	9
12	14 17	9 15	4 4	Min(9-4;15-4)	5
11	12	5	0	5-0	5
10	11	5	1	5-1	4
8	14 17	9 15	6 4	Min(9-6;15-4)	3
7	8	3	0	3-0	3
6	7	3	1	3-1	2
5	6	2	2	2-2	0
4	5	1	1	1-1	0
3	5	1	1	1-1	0
1	3 4 10	1 3 4	1 3 2	Min(1-1;3-3;4-2)	0

На основе найденных значений раннего и позднего времени начала работ можно найти критический путь. Соответствующие вычисления представлены в таблице 7.

Таблица 7 – Вычисление критического пути

Работа №	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	0	0
4	0	0	0
5	3	3	0
6	2	2	0
7	3	3	0
8	8	8	0
10	0	4	4
11	2	5	3
12	3	5	2
14	8	8	0
15	12	12	0
16	14	16	2
17	8	15	7
20	14	14	0
21	18	18	0
22	18	38	20
23	38	38	0

Продолжение таблицы 7

Работа №	Раннее время начала	Позднее время начала	Резерв времени
25	38	41	3
26	41	41	0
27	44	44	0
28	51	51	0
29	51	51	0

На основе полученных можно сделать вывод, что критический путь проходит через работы с номерами 1, 3, 4, 6, 5, 6, 7, 8, 14, 15, 20, 21, 23, 26, 27, 28, 29, а общее время выполнения проекта составляет 51 день.

На рисунке 17 представлен граф проекта, на котором красными линиями указан критический путь.

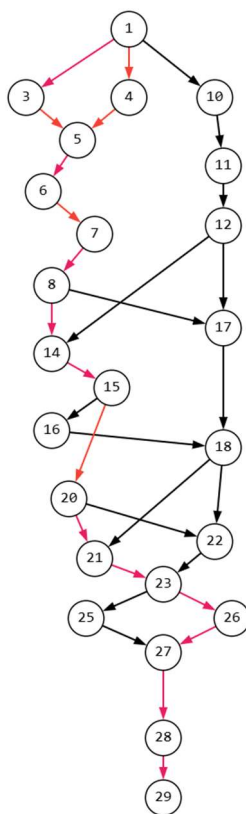


Рисунок 17 – Критический путь проекта

Теперь проверим правильность расчетов продолжительности проекта и вычислим величину его бюджета. Для этого воспользуемся системой управления проектами Microsoft Project 2007, которая позволяет автоматически произвести все необходимые вычисления [15]. Список составленных задач в системе

Microsoft Project представлен на рисунке 18. В качестве времени начала проекта была выбрана дата 8.03.2025.

	Название задачи	Длительность	Начало	Окончание	Предш.	Названия ресурсов
1	Начало проекта	0 дней	Сб 08.03.25	Сб 08.03.25		
2	Анализ требований	10 дней	Пн 10.03.25	Пт 21.03.25		
3	Анализ предметной области и пост	1 день	Пн 10.03.25	Вт 11.03.25	1	Аналитик[40%];Компьютер[1];Интернет[200,00 Р]
4	Анализ программных продуктов а	3 дней	Пн 10.03.25	Чт 13.03.25	1	Аналитик[50%];Компьютер[1];Интернет[200,00 Р];Смартфон[1]
5	Анализ функциональных требовани	2 дней	Пт 14.03.25	Пн 17.03.25	3,4	Аналитик[50%];Компьютер[1];Интернет[150,00 Р];Программист1;Программист2
6	Анализ требований к интерфейсу	2 дней	Вт 18.03.25	Чт 20.03.25	5	Аналитик[30%];Дизайнер;Интернет[200,00 Р];Компьютер[2]
7	Анализ целевой аудитории	1 день	Пт 21.03.25	Пт 21.03.25	6	Аналитик[50%];Интернет;Компьютер[1]
8	Анализ требований завершен	0 дней	Пт 21.03.25	Пт 21.03.25	7	
9	Выбор средств разработки ПО	5 дней	Пт 21.03.25	Пт 28.03.25		
10	Анализ существующих технологий	2 дней	Пт 21.03.25	Чт 27.03.25	8	Аналитик[0%];Программист1[30%];Программист2[0%];Компьютер[1];Интернет[200,00 Р]
11	Выбор технологий разработки ПО	1 день	Пт 28.03.25	Пт 28.03.25	10	Программист1[50%];Аналитик[50%];Компьютер[2]
12	Выбор средств разработки ПО за	0 дней	Пт 28.03.25	Пт 28.03.25	11	
13	Проектирование ПО	11,5 дней	Пн 31.03.25	Вт 15.04.25		
14	Проектирование архитектуры прил	3 дней	Пн 31.03.25	Ср 02.04.25	12	Программист1[50%];Программист2;Компьютер[2];Аналитик[50%]
15	Проектирование БД	1 день	Чт 03.04.25	Чт 03.04.25	14	Компьютер[1];Программист2[70%];Аналитик[50%]
16	Проектирование серверной части г	4 дней	Пт 04.04.25	Ср 09.04.25	15	Программист1[50%];Аналитик[30%];Компьютер[2];Интернет
17	Проектирование интерфейса польз	2 дней	Чт 10.04.25	Вт 15.04.25	16	Дизайнер[60%];Компьютер[2];Интернет[300,00 Р];Программист1[30%];Аналитик[30%]
18	Проектирование приложения завер	0 дней	Вт 15.04.25	Вт 15.04.25	17,16	
19	Реализация ПО	22 дней	Вт 15.04.25	Чт 15.05.25		
20	Реализация БД	7 дней	Вт 15.04.25	Вт 29.04.25	18	Программист1[0%];Программист2;Компьютер[2];Интернет[300,00 Р]
21	Реализация серверной части	12 дней	Вт 29.04.25	Чт 15.05.25	18,20	Программист1[40%];Компьютер[1]
22	Реализация мобильного приложени	10 дней	Вт 29.04.25	Вт 13.05.25	18,20	Программист2[50%];Дизайнер[20%];Компьютер[2];Программист1[60%];Смартфон[1]
23	Реализация завершена	0 дней	Чт 15.05.25	Чт 15.05.25	21,22	
24	Тестирование	4,29 дней	Чт 15.05.25	Ср 21.05.25		
25	Функциональное тестирование	4,2 дней	Чт 15.05.25	Ср 21.05.25	23	Компьютер[1];Интернет[100,00 Р];Тестировщик[50%]
26	Модульное тестирование	3 дней	Чт 15.05.25	Вт 20.05.25	23	Программист2[50%];Компьютер[1];Тестировщик[50%]
27	Тестирование завершено	0 дней	Ср 21.05.25	Ср 21.05.25	25,26	
28	Составление и оформление документ	1 день	Ср 21.05.25	Чт 22.05.25	27	Аналитик;Программист1[50%];Программист2[50%];Компьютер[3];Интернет
29	Конец проекта	0 дней	Чт 22.05.25	Чт 22.05.25	28	

Рисунок 18 – Список задач проекта в системе Microsoft Project

Построенная системой Microsoft Project диаграмма Ганта приведена на рисунке 19.

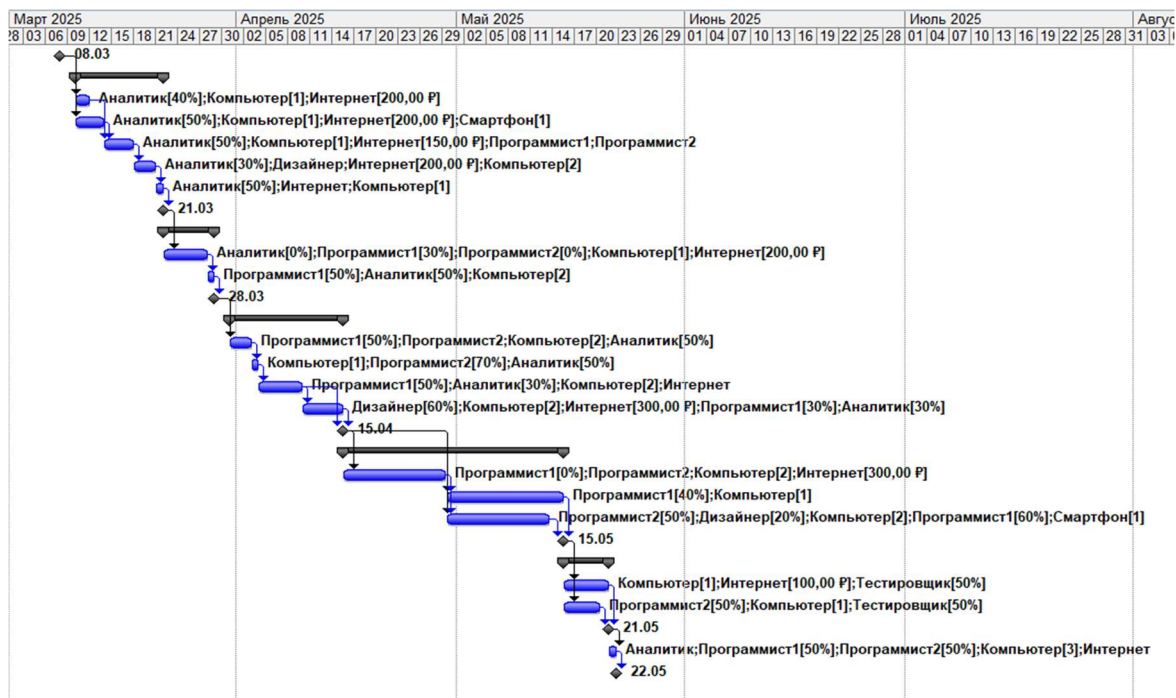


Рисунок 19 – Диаграмма Ганта

График загрузки ресурса «Программист 1» представлен на рисунке 20.

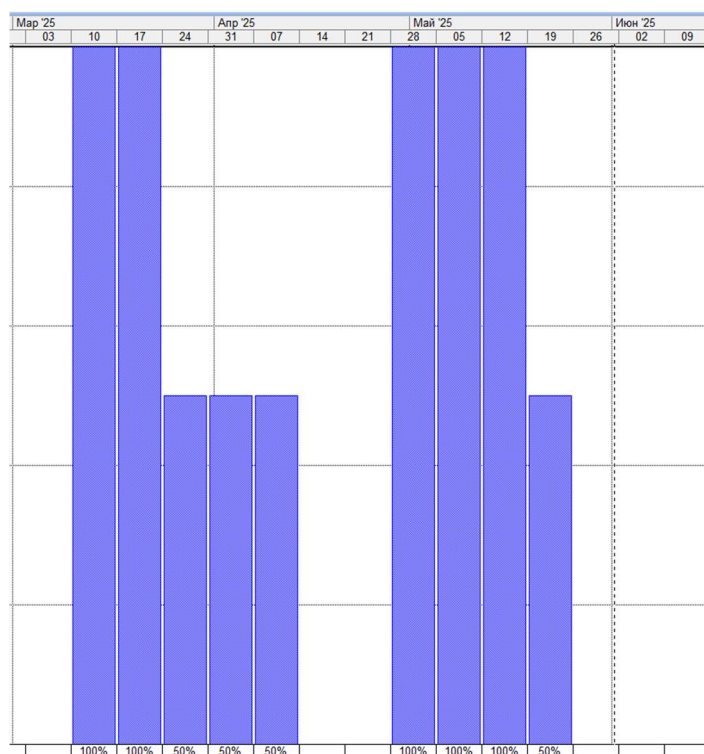


Рисунок 20 – График загрузки ресурса «Программист 1»

График загрузки ресурса «Программист 2» представлен на рисунке 21.

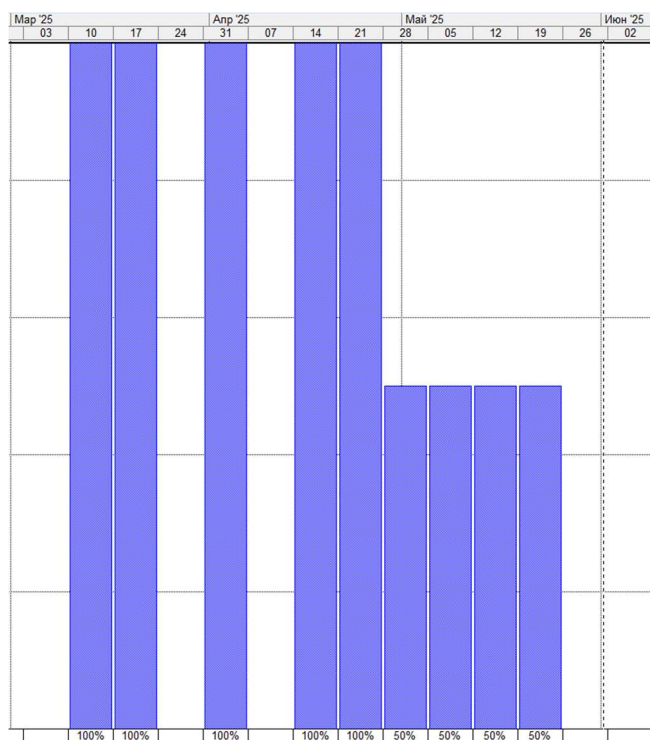


Рисунок 21 – График загрузки ресурса «Программист 2»

График загрузки ресурса «Аналитик» представлен на рисунке 22.

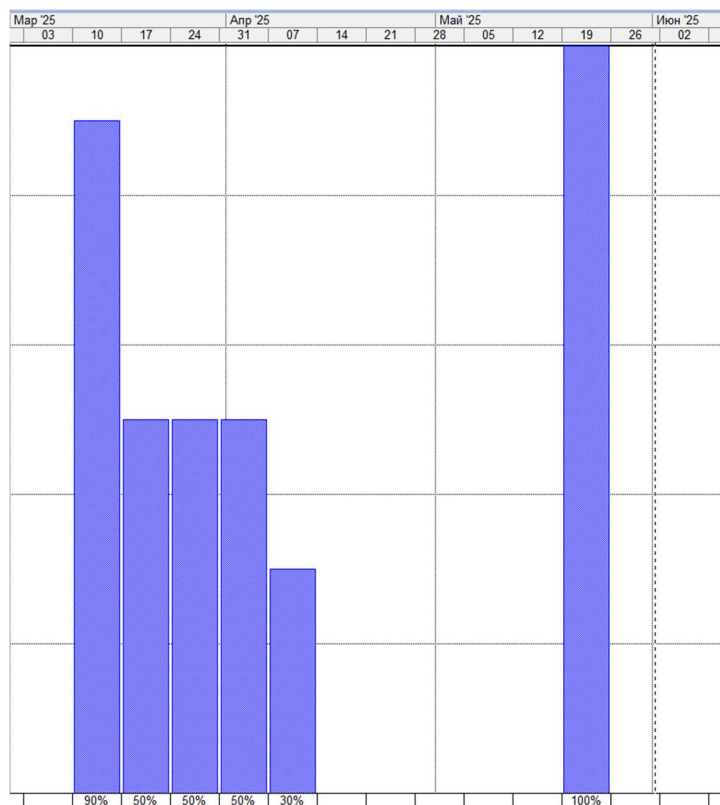


Рисунок 22 – График загрузки ресурса «Аналитик»

График загрузки ресурса «Дизайнер» представлен на рисунке 23.

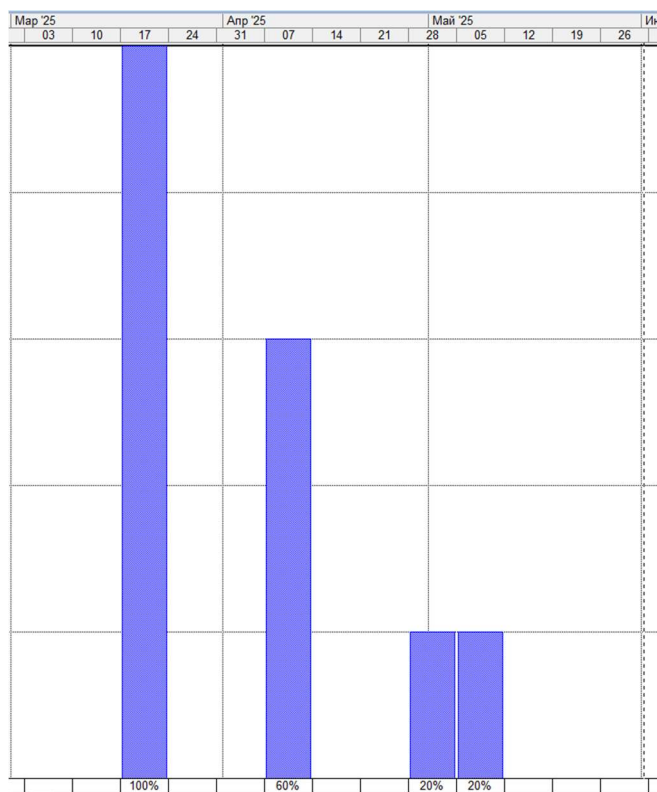


Рисунок 23 – График загрузки ресурса «Дизайнер»

График загрузки ресурса «Тестировщик» представлен на рисунке 24.

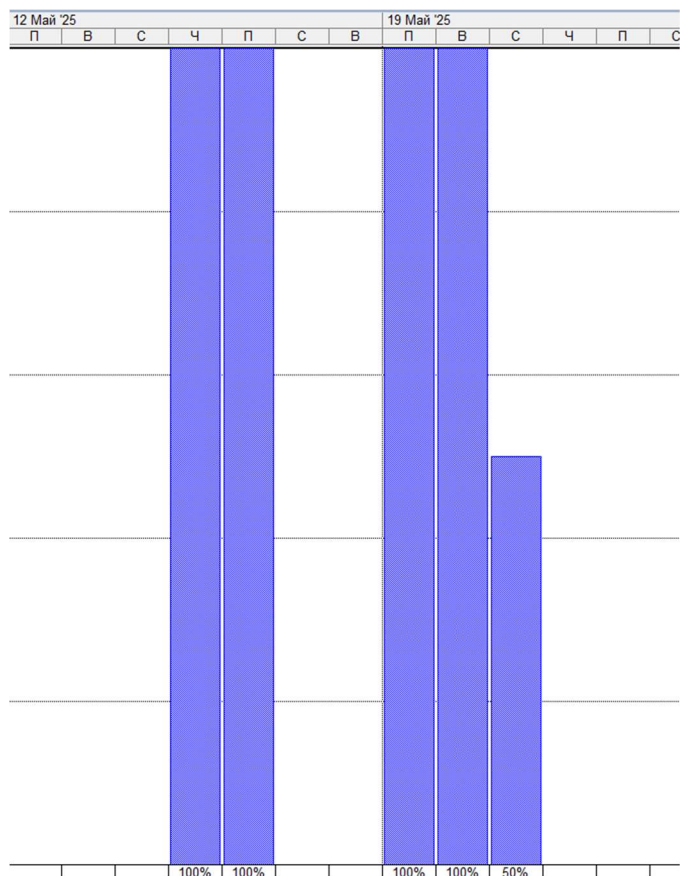


Рисунок 24 – График загрузки ресурса «Тестировщик»

На рисунке 25 представлен обзор проекта. Из рисунка видно, что предполагаемые затраты составляют 206 450 руб.

Даты			
Начало:	Сб 08.03.25	Окончание:	Вт 20.05.25
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	51,79 дней	Оставшаяся:	51,79 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	51,79 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	410,82 ч	Оставшиеся:	410,82 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	410,82 ч	Процент завершения:	0%
Затраты			
Текущий план:	206 450,00 Р	Оставшиеся:	206 450,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	206 450,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	29	Трудовые ресурсы:	6
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дост	0
Завершённые задачи:	0	Материальные ресурсы:	2
Итого задач:	29	Итого ресурсов:	8

Рисунок 25 – Обзорный отчёт проекта

На основе произведённых оценок стоимости и сроков проекта можно сделать вывод о его экономической целесообразности.

2 Проектирование программного обеспечения

2.1 Выбор технологий и инструментальных средств разработки

В ходе проектирования программного обеспечения была выбрана широко распространенная методология разработки Rational Unified Process (RUP). В первую очередь, данная методология характеризуется итеративным принципом создания ПО. Вся разработка делится на 4 основных фазы, каждая из которых может состоять из нескольких итераций [16]:

- начальная фаза – определение границ проекта, формулировка ключевых требований, экономическое обоснование;
- фаза уточнения – анализ предметной области, актуализация экономического обоснования, проектирование архитектуры;
- фаза построения – реализация основной части функциональности продукта;
- фаза внедрения – доработка продукта до окончательной версии, передача готового решения заказчику и обучение конечных пользователей.

Выбор технологии RUP обусловлен рядом преимуществ относительно других методологий. Ключевым достоинством RUP является обеспечение гибкости процесса разработки, позволяющей относительно быстро вносить корректировки в случае обнаружения ошибок на предыдущих этапах или изменений требований со стороны заказчика. Важной особенностью данной методологии является оценка рисков уже на первых этапах разработки. Это дает возможность завершить проект еще на начальном этапе в случае его экономической нецелесообразности. Кроме того, в отличие от других методологий, таких как eXtreme Programming (XP), RUP не требует постоянного взаимодействия с заказчиком.

Для анализа требований было решено использовать диаграммы на языке моделирования UML. Для построения диаграмм использовалось средство UML-моделирования Visual Paradigm.

В качестве системы управления базами данных была выбрана Microsoft SQL Server 2019, которая является одной из самых популярных СУБД [17]. Реализация базы данных будет производиться с использованием языка SQL. Серверная часть приложения будет использовать технологию Entity Framework Core для работы с базой данных. Данный инструмент позволяет представлять сущности базы данных в виде классов C#, а также предоставляет возможность выполнять SQL запросы через язык запросов Linq [18].

Серверная часть приложения будет реализована при помощи платформы ASP.NET. Данная платформа предоставляет широкий функционал для обработки HTTP-запросов и методы управления поведением сервера. Сервер будет работать под управлением операционной системы Windows 10.

Клиентская часть приложения будет реализована при помощи платформы .NET MAUI, которая предназначена для построения приложений для iOS и Android. Целевой платформой в данной разработке будет являться Android.

2.2 Проектирование архитектуры программного обеспечения

В данной работе было принято решение использовать трехуровневую архитектуру «клиент-сервер». Схема архитектуры представлена на рисунке 26.

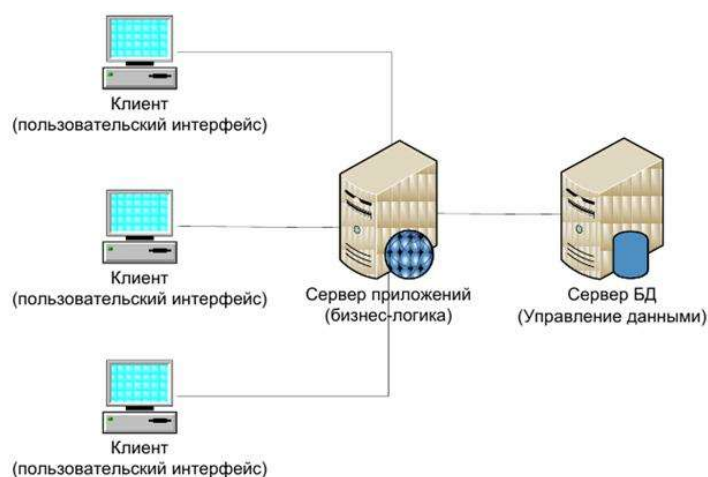


Рисунок 26 – Трехуровневая архитектура «клиент-сервер»

В данной архитектуре каждый компонент выполняет строго определенные функции. Уровень управления данными (сервер базы данных) обеспечивает надежное хранение информации, выполняет CRUD операции (операции

создания, чтения, обновления и удаления) и отвечает за целостность данных. Уровень бизнес-логики (сервер приложений) реализует основной функционал системы, обрабатывает запросы клиента и взаимодействует с СУБД. Уровень пользовательского интерфейса (клиент) предоставляет интерфейс для взаимодействия с пользователем, отображает информацию в удобном для человека формате и обрабатывает действия пользователя. В данной работе в качестве клиентских терминалов будут использоваться мобильные устройства.

Ключевыми достоинствами архитектуры являются разделение ответственности и параллельная разработка, что обусловлено независимостью уровней, гибкость масштабирования и устойчивость к изменениям, а также повышенная безопасность.

2.3 Проектирование базы данных

2.3.1 Концептуальное проектирование базы данных

На основе анализа предметной области была построена концептуальная модель базы данных. Графическое представление концептуальной модели БД в виде диаграммы «сущность-связь» представлено на рисунке 27.

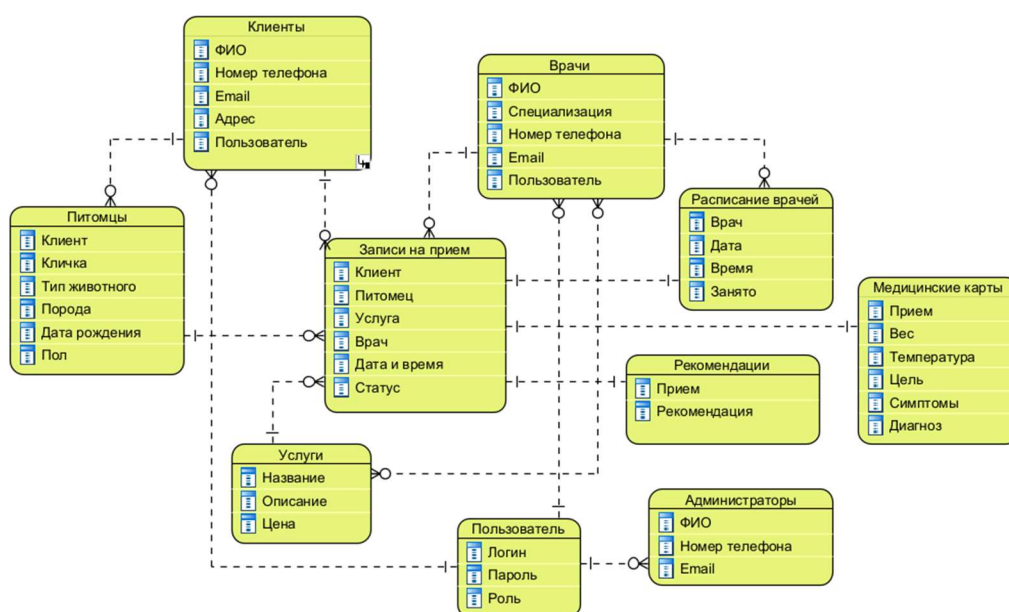


Рисунок 27 – Графическое представление концептуальной модели базы данных

Сущности концептуальной модели данных представлены в таблице 8.

Таблица 8 – Сущности концептуальной модели

№	Название	Описание	Количество экземпляров	Режим обновления
1	Пользователи	Список пользователей системы	10000	При регистрации нового пользователя
2	Клиенты	Список клиентов ветеринарной клиники	10000	При регистрации клиента
3	Врачи	Список врачей ветеринарной клиники	50	При принятии на работу нового врача
4	Администратор	Список администраторов ветеринарной клиники	20	При принятии на работу нового администратора
5	Питомцы	Список питомцев, посещающих ветеринарную клинику	10000	При записи на прием нового питомца
6	Услуги	Список услуг ветеринарной клиники	500	При появлении новой услуги или изменении цен
7	Расписание врачей	Список слотов для записи к ветеринарному врачу	30000	При добавлении нового слота
8	Записи на прием	Список записей на прием	30000	При новой записи к врачу или изменении старой
9	Рекомендации	Список рекомендаций ветеринарного врача	10000	При добавлении новой рекомендации после приема
10	Медицинские карты	Медицинские записи ветеринарного врача о приеме	30000	При посещении ветеринарного врача на приеме

В таблице 9 представлено описание связей между сущностями.

Таблица 9 – Описание связей между сущностями концептуальной модели

№	Связь	Описание	Мощность	Обязательность
1	Пользователь – Клиент	Определяет роль пользователя	1:M	Да
2	Пользователь – Врач	Определяет роль пользователя	1:M	Да
3	Пользователь – Администратор	Определяет роль пользователя	1:M	Да
4	Клиенты – Питомцы	Определяет питомцев клиента	1:M	Да
5	Врачи – Расписание врачей	Определяет свободные места для записи к ветеринару	1:M	Да
6	Клиенты – Записи на прием	Определяет записи клиента	1:M	Да
7	Питомцы – Записи на прием	Определяет питомца, записанного на прием	1:M	Да
8	Услуга – Записи на прием	Определяет услугу, на которую записался клиент	1:M	Нет
9	Врачи – Записи на прием	Определяет врача, к которому записался клиент	1:M	Да
10	Рекомендации – Записи на прием	Определяет рекомендации, оставленные врачом после приема	1:1	Нет
11	Врачи – Услуги	Определяет, какие услуги предоставляются какими врачами	M:M	Да
12	Медицинские карты – Записи на прием	Определяет прием медицинской карты	1:1	Да

Продолжение таблицы 10

5	Admins	Список администраторов ветеринарной клиники	20	При принятии на работу нового администратора
6	Pets	Список питомцев, посещающих ветеринарную клинику	10000	При записи на прием нового питомца
7	VetServices	Список услуг ветеринарной клиники	500	При появлении новой услуги или изменении цен
8	DoctorSchedules	Список слотов для записи к врачу	30000	При добавлении нового слота
9	Appointments	Список записей на прием	30000	При новой записи к врачу или изменении старой
10	Recommendations	Список рекомендаций врача	10000	При добавлении новой рекомендации после приема
11	MedicalRecords	Медицинские записи ветеринарного врача о приеме	30000	При посещении ветеринарного врача на приеме
12	Status	Список возможных статусов приема	3	При проектировании системы
13	Diagnosis	Список диагнозов по ГОСТу	2000	При проектировании системы

Также в таблице 11 представлено описание связей между сущностями логической модели, не определенных в результате концептуального моделирования.

Таблица 11 – Описание связей между сущностями логической модели

№	Связь	Описание	Мощность	Обязательность
1	Users – Clients	Определяет роль пользователя	1:M	Да
2	Users – Doctors	Определяет роль пользователя	1:M	Да
3	Users – Admins	Определяет роль пользователя	1:M	Да
4	Clients – Pets	Определяет питомцев клиента	1:M	Да
5	Doctors – DoctorSchedules	Определяет свободные места для записи к ветеринару	1:M	Да
6	Clients – Appointments	Определяет записи клиента	1:M	Да
7	Pets – Appointments	Определяет питомца, записанного на прием	1:M	Да
8	VetServices – Appointments	Определяет услугу, на которую записался клиент	1:M	Нет
9	Doctors – Appointments	Определяет врача, к которому записался клиент	1:M	Да
10	Recommendations – Appointments	Определяет рекомендации, оставленные врачом после приема	1:1	Нет
11	DoctorSchedules – Appointments	Определяет дату и время записи	1:1	Да
12	UserRole – User	Определяет роль пользователя	1:1	Да
13	Appointments – Status	Определяет статус приема	1:1	Да
14	MedicalRecords – Appointments	Определяет прием медицинской карты	1:1	Да
15	MedicalRecords – Diagnosis	Определяет диагнозы, доступные для указания в медицинской карте	1:1	Да

Несмотря на достаточную содержательность данной модели, сущности все еще представлены в абстрактной форме, без привязки к конкретной реализации базы данных [19].

2.4 Проектирование серверной части приложения

Серверная часть приложения реализована в виде REST API, где взаимодействие между клиентом и сервером осуществляется по протоколу HTTP с использованием стандартных методов, таких как GET (получение данных ресурса), PUT (обновление существующего ресурса), POST (создание нового ресурса) и DELETE (удаление ресурса). Доступ к ресурсам сервера осуществляется через URI-идентификаторы.

Для реализации приложения был выбран шаблон MVC (Model-View-Controller). Схема данного шаблона приведена на рисунке 29.

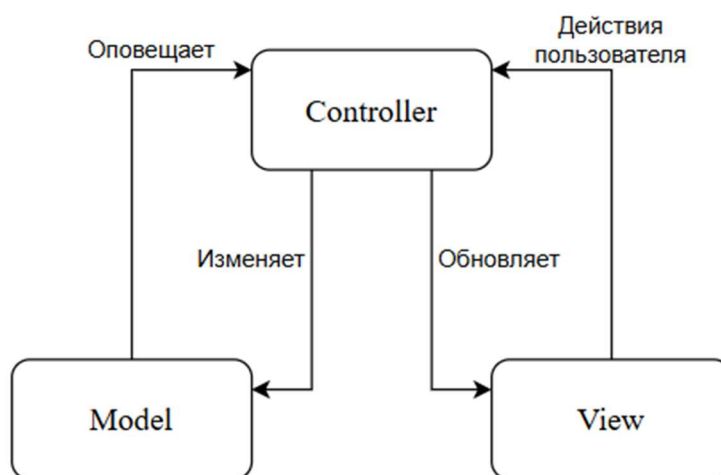


Рисунок 29 – Схема шаблона MVC

Как видно из данной схемы, шаблон включает 3 ключевых компонента: модель (Model), контроллер (Controller) и представление (View) [20]. Классы уровня модели обеспечивают доступ к данным и взаимодействие с базой данных. Классы уровня контроллера отвечают за организацию взаимодействия пользователя с клиентом, обрабатывают HTTP-запросы и формируют HTTP-ответы. Уровень представления отвечает за взаимодействие с пользователем.

Выбор данного шаблона обусловлен использованием в работе технологии ASP.NET, которая нацелена на применение данного шаблона.

2.5 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был спроектирован интерфейс мобильного приложения. Приложение включает в себя 3 набора интерфейса в зависимости от роли пользователя. Макет интерфейса страницы авторизации представлен на рисунке 30.

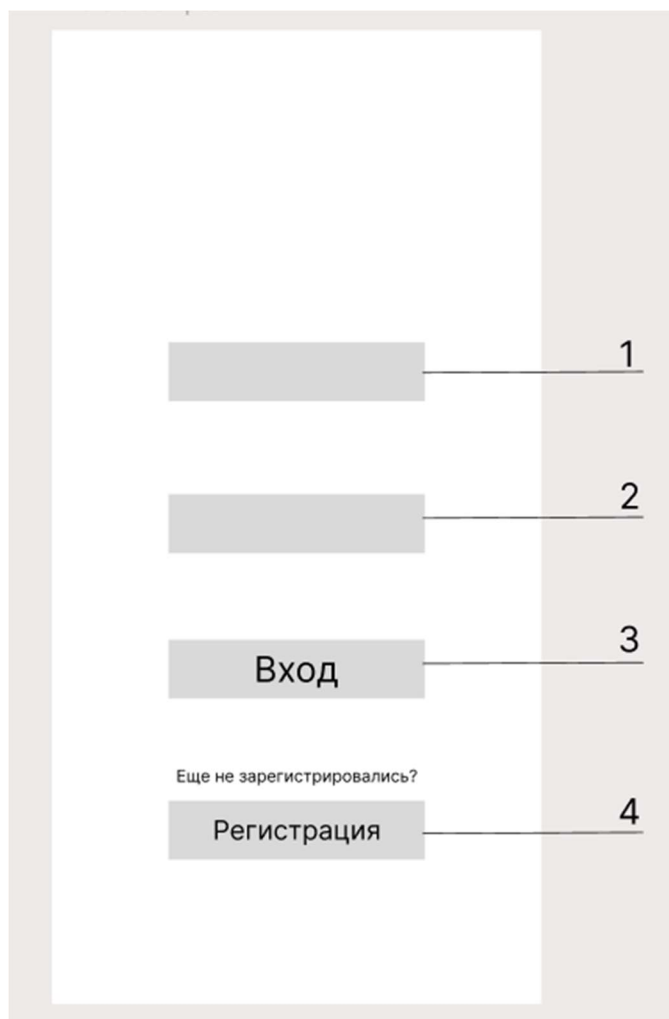


Рисунок 30 – Страница авторизации

На макете страницы авторизации имеются следующие элементы:

- 1 – поле ввода логина;
- 2 – поле ввода пароля;
- 3 – кнопка входа, по нажатию на которую осуществляется попытка авторизации;
- 4 – кнопка регистрации, по нажатию которой открывается страница регистрации.

Макет интерфейса страницы регистрации представлен на рисунке 31.

The image shows a wireframe of a registration page. It consists of a central white rectangular area with a light gray border, set against a light pink background. Inside the white area, there are seven input fields stacked vertically, each with a label to its left. Below these fields are two buttons: 'Регистрация' (Registration) and 'Вход' (Login). A link 'Уже зарегистрировались?' (Already registered?) is positioned between the two buttons. Numbered lines (1-9) point to the following elements: 1. 'Фамилия' (Surname) input field; 2. 'Имя' (Name) input field; 3. 'Отчество' (Patronymic) input field; 4. 'Номер телефона' (Phone number) input field; 5. 'email' input field; 6. 'Логин' (Login) input field; 7. 'Пароль' (Password) input field; 8. 'Регистрация' button; 9. 'Вход' button.

Рисунок 31 – Страница регистрации

На макете страницы регистрации имеются следующие элементы:

- 1 – поле ввода фамилии пользователя;
- 2 – поле ввода имени пользователя;
- 3 – поле ввода отчества пользователя;
- 4 – поле ввода номера телефона пользователя;
- 5 – поле ввода email пользователя;
- 6 – поле ввода логина пользователя;
- 7 – поле ввода пароля пользователя;
- 8 – кнопка регистрации, по нажатию на которую осуществляется попытка регистрации;
- 9 – кнопка входа, по нажатию которой открывается страница входа.

После авторизации пользователь переходит на страницы клиента, врача или администратора, в зависимости от роли. Клиенту доступны страницы просмотра запланированных и прошедших приемов, услуг и цен ветеринарной клиники, записи на прием и профиля. Макет интерфейса страницы просмотра услуг и цен представлен на рисунке 32.

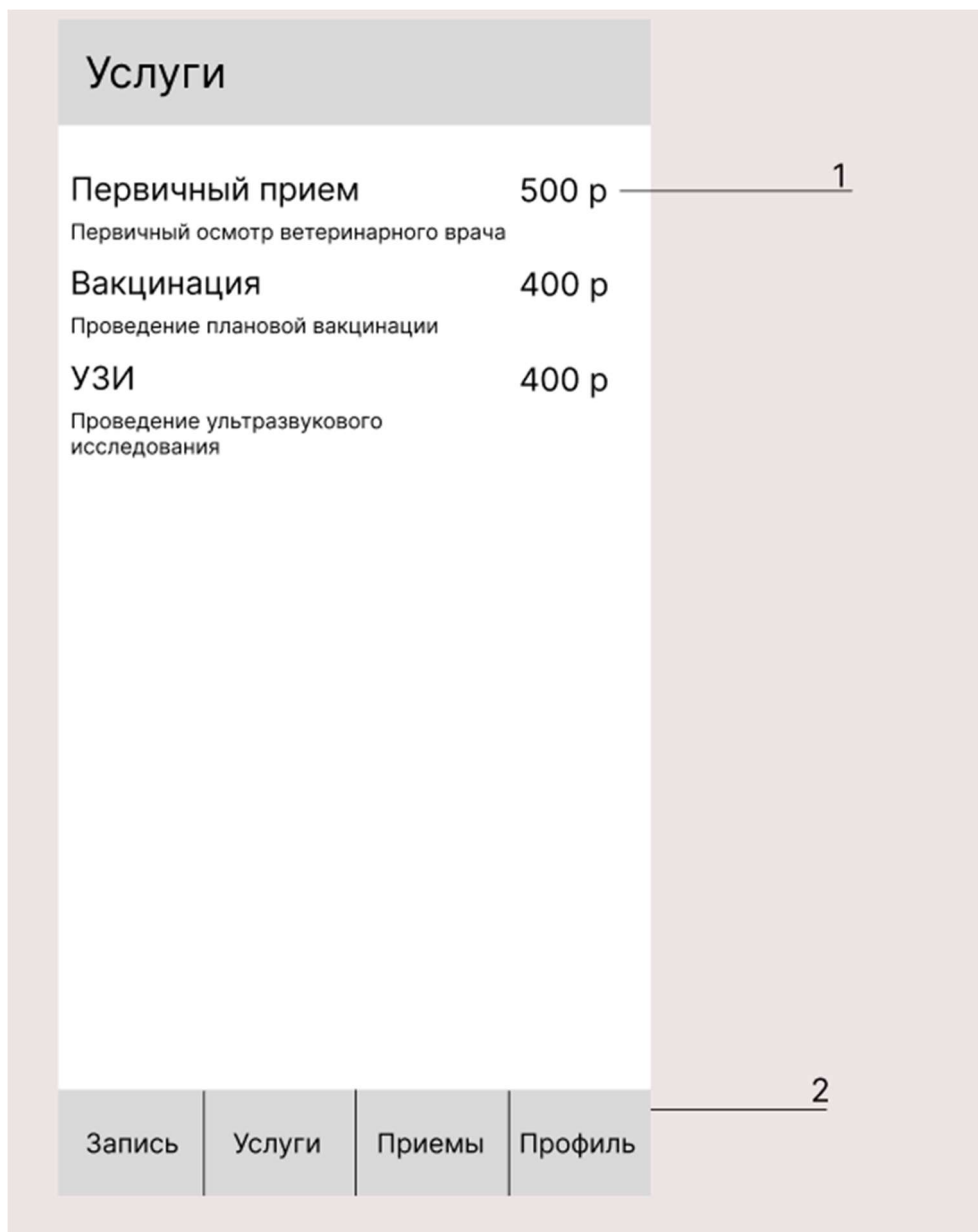


Рисунок 32 – Страница просмотра услуг и цен

На макете страницы услуг имеются следующие элементы:

- 1 – список услуг, их описания и цен на услуги;
- 2 – панель навигации.

Макет интерфейса страницы записи к ветеринарному врачу представлен на рисунке 33.

The mockup shows a page titled "Запись" (Appointment) with five dropdown menus for selection. Each dropdown is represented by a gray rectangle with a downward arrow icon. To the right of each dropdown is a number (1-5) with a line pointing to the arrow icon. The dropdowns are labeled "Услуга" (Service), "Врач" (Doctor), "Дата" (Date), "Время" (Time), and "Питомец" (Pet). At the bottom, there is a navigation bar with four buttons: "Запись", "Услуги", "Приемы", and "Профиль".

Услуга	Врач	Дата	Время	Питомец

Запись	Услуги	Приемы	Профиль
--------	--------	--------	---------

Рисунок 33 – Страница записи к ветеринарному врачу

На макете страницы записи к ветеринарному врачу имеются следующие элементы:

- 1 – выпадающий список для выбора услуги;
- 2 – выпадающий список для выбора врача;
- 3 – выпадающий список для выбора даты;
- 4 – выпадающий список для выбора времени;
- 5 – выпадающий список для выбора питомца.

Макет интерфейса страницы запланированных и прошедших приемов клиента представлен на рисунке 34.

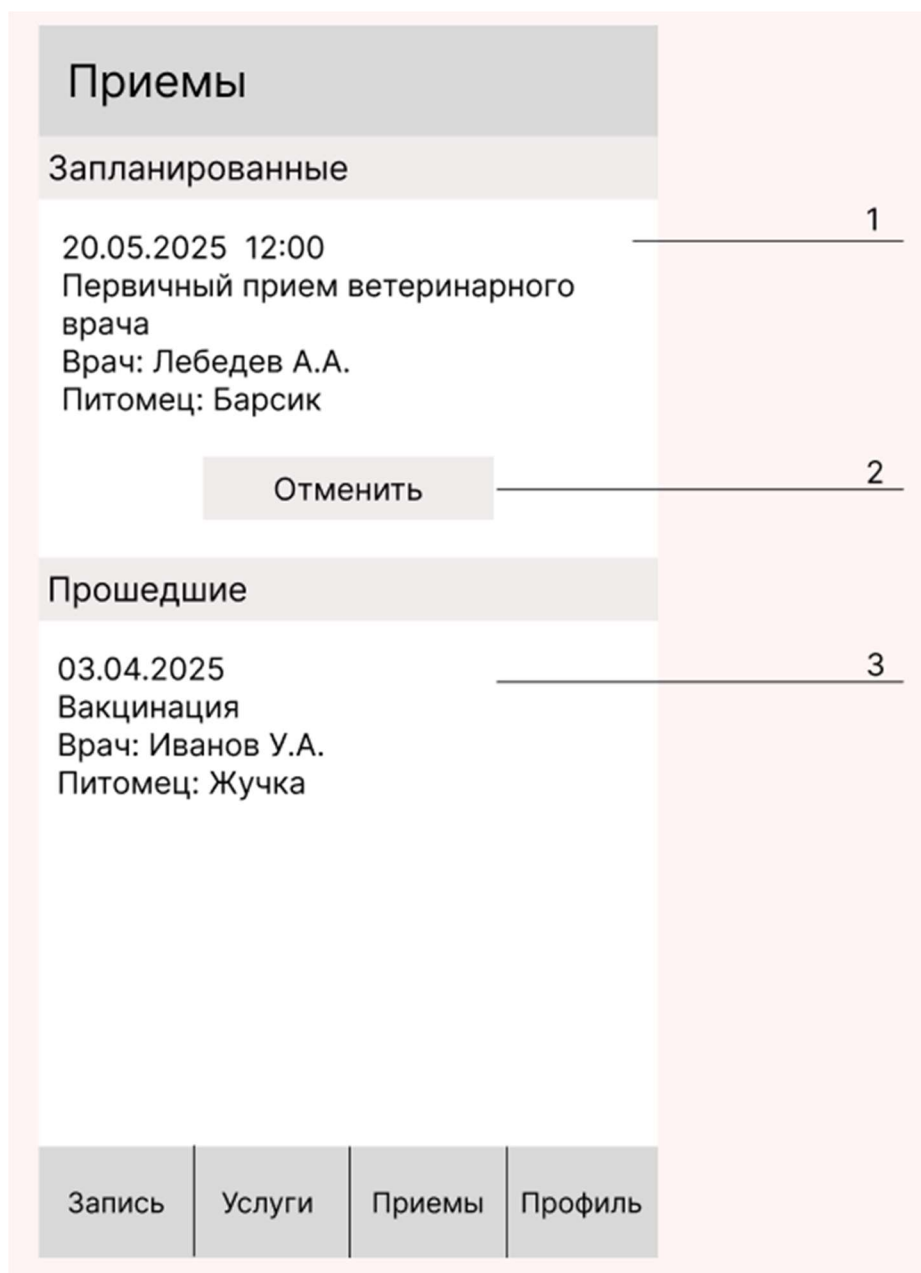


Рисунок 34 – Страница приемов

На макете страницы приемов имеются следующие элементы:

- 1 – список запланированных приемов, с указанием даты и времени записи, услуги, врача и питомца;
- 2 – кнопка отмены записи, отменяющая запись на приеме;
- 3 – список прошедших приемов с указанием даты, услуги, врача и питомца.

По нажатию на прием открывается страница просмотра рекомендаций после выбранного приема. Макет интерфейса страницы рекомендаций представлен на рисунке 35.

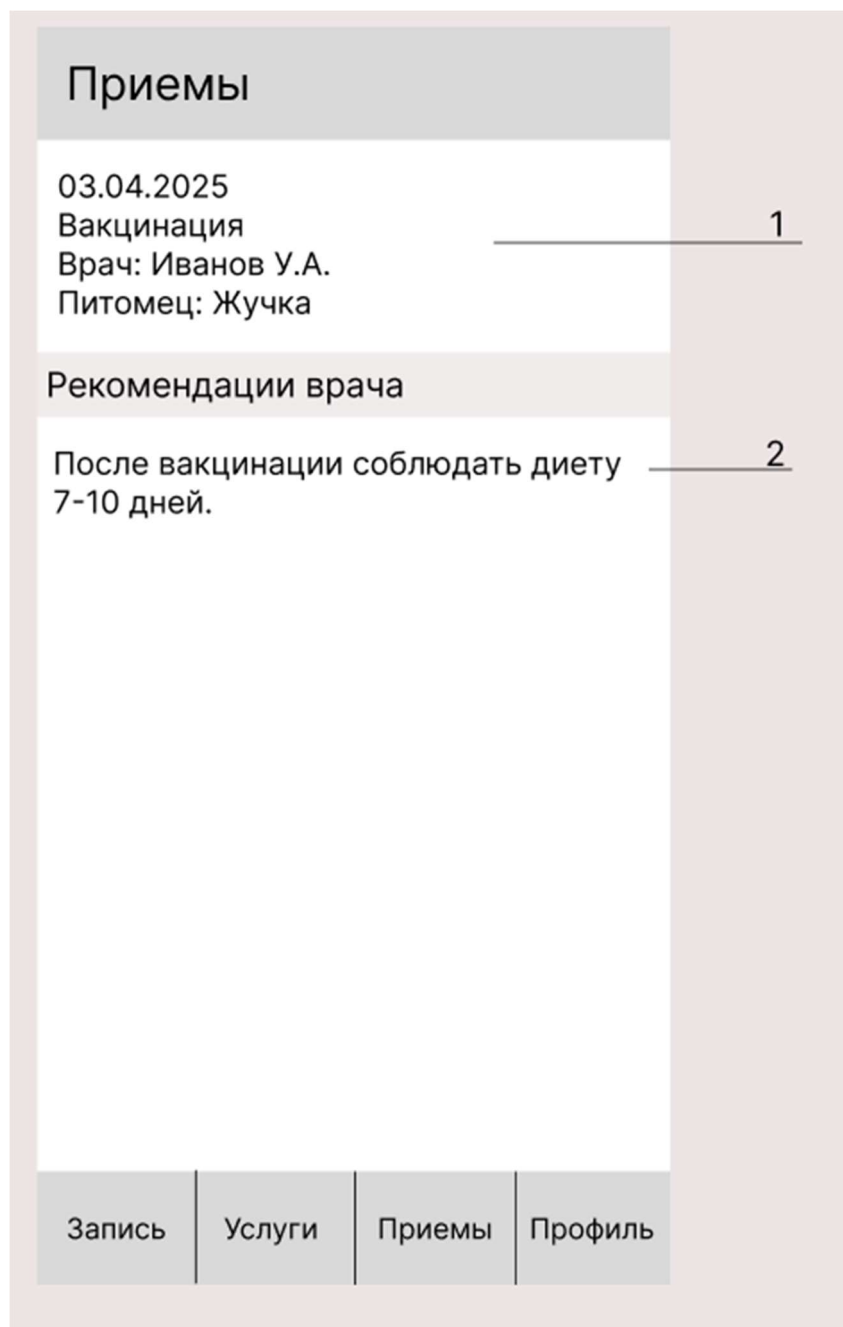


Рисунок 35 – Страница рекомендаций после приема

На макете страницы рекомендаций имеются следующие элементы:

1 – выбранный прием с указанием даты и времени записи, услуги, врача и питомца;

2 – рекомендация ветеринарного врача после приема.

Макет интерфейса страницы профиля представлен на рисунке 36.

The mockup shows a user profile page with a light gray header and a white main content area. The header contains the title 'Профиль'. The main content area is divided into three sections: 1. Profile information: 'Иванов Игорь Петрович', '89270573541', and 'igorivanov@gmail.com'. 2. Pets: 'Жучка' and 'Барсик'. 3. A button 'Добавить питомца'. A bottom navigation bar contains four buttons: 'Запись', 'Услуги', 'Приемы', and 'Профиль'. Numbered callouts 1, 2, and 3 point to the profile information, the list of pets, and the 'Добавить питомца' button respectively.

Профиль			
Иванов Игорь Петрович		1	
89270573541			
igorivanov@gmail.com			
Питомцы			
Жучка		2	
Барсик			
Добавить питомца		3	

Запись	Услуги	Приемы	Профиль
--------	--------	--------	---------

Рисунок 36 – Страница профиля

На макете страницы профиля имеются следующие элементы:

- 1 – информация о клиенте;
- 2 – список питомцев, по нажатию на питомца откроется страница редактирования питомца;
- 3 – кнопка добавления питомца, по нажатию на которую откроется страница добавления нового питомца.

Макет интерфейса страницы добавления питомца представлен на рисунке 37.

Профиль

Кличка 1

Животное 2

Порода 3

Дата рождения 4

Пол 5

Добавить питомца 6

Запись Услуги Приемы Профиль

Рисунок 37 – Страница добавления питомца

На макете страницы добавления питомца имеются следующие элементы:

- 1 – поле ввода клички питомца;
- 2 – поле ввода названия животного;
- 3 – поле ввода породы питомца;
- 4 – поле ввода даты рождения питомца;
- 5 – поле ввода пола питомца;
- 6 – кнопка добавления питомца, по нажатию на которую осуществляется попытка добавления питомца.

После авторизации пользователя как ветеринарный врач, пользователю доступны страницы просмотра запланированных и прошедших приемов,

добавления расписания, поиска питомца по клиентам и профиль. Макет интерфейса страницы приемов ветеринарного врача представлен на рисунке 38.

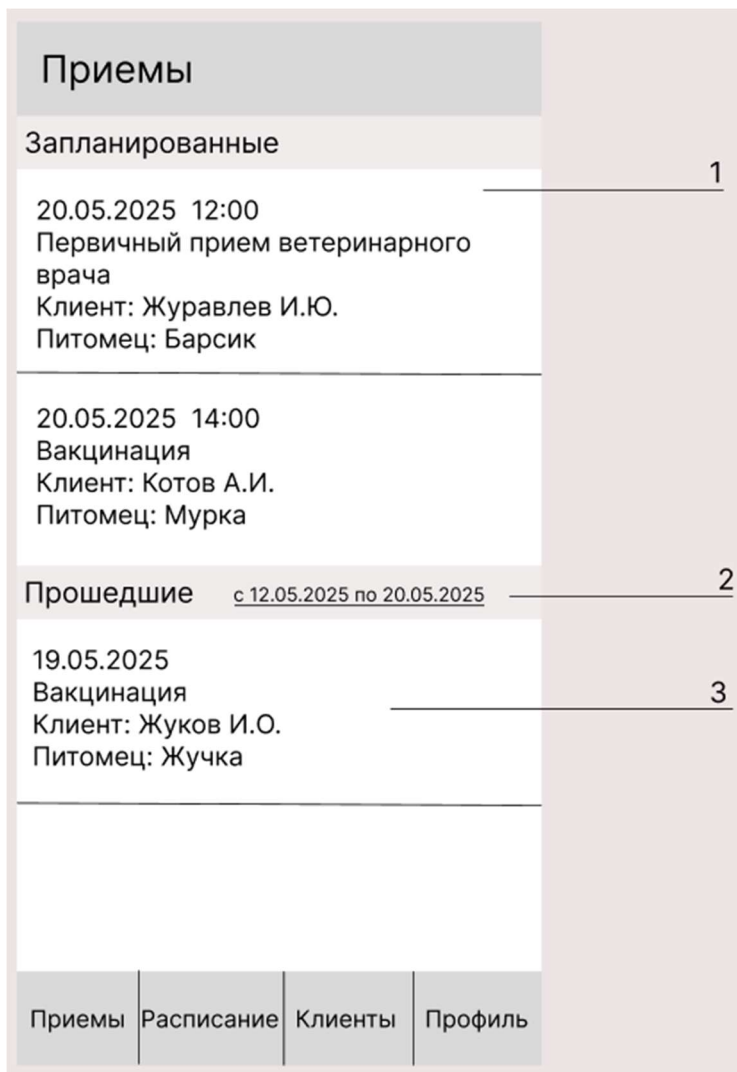


Рисунок 38 – Страница приемов ветеринарного врача

На макете страницы приемов ветеринарного врача имеются следующие элементы:

- 1 – список запланированных приемов в порядке очередности по времени приема, по нажатию на прием открывается подробная информация и медицинская карточка питомца;
- 2 – фильтр для прошедших приемов;
- 3 – список прошедших приемов, по нажатию на которые открывается подробная информация о приеме и медицинская карточка питомца.

Макет интерфейса страницы приема и медицинской карточки представлен на рисунке 39.

Прием

20.05.2025 12:00
 Первичный прием ветеринарного врача
 Клиент: Журавлев И.Ю.
 Питомец: Кот Барсик
 Дата рождения: 12.07.2020
 Порода: Мейн-кун
 Пол: мужской

История посещений

Вес Температура

Цель посещения

Жалобы, симптомы

Диагноз

Рекомендации

Приемы | Расписание | Клиенты | Профиль

Рисунок 39 – Страница приема и медицинской карточки питомца

На макете страницы приема питомца имеются следующие элементы:

- 1 – основная информация о приеме: дата и время, клиент, информация о питомце;
- 2 – кнопка истории посещений, по нажатию на которую откроется страница списка прошедших посещений питомца;
- 3 – поля ввода веса и температуры питомца;
- 4 – поле ввода цели посещения ветеринарного врача;
- 5 – поле ввода жалоб клиента и симптомов;
- 6 – поле ввода диагноза;
- 7 – поле ввода рекомендаций для лечения.

Макет интерфейса страницы поиска питомца по клиенту представлен на рисунке 40.

Клиенты

Клиент: Журавлев Иван 1

Барсик
Кот 2
Дата рождения: 12.07.2020
Порода: Мейн-кун
Пол: мужской

Пончик
Собака
Дата рождения: 19.09.2022
Порода: Померанский щпиц
Пол: мужской

Приемы | Расписание | Клиенты | Профиль

Рисунок 40 – Страница поиска питомцев по клиенту

На макете страницы поиска питомцев по клиенту имеются следующие элементы:

- 1 – поле для выбора клиента;
- 2 – список питомцев клиента, по нажатию на которых открывается страница редактирования питомца и история посещений.

Макет интерфейса страницы редактирования питомца и истории посещений представлен на рисунке 41.

Питомец

Клиент: Журавлев И.Ю.
 Кличка: Барсик
 Животное: Кот
 Порода: Мейн-кун
 Дата рождения: 12.07.2020
 Пол: мужской

Приемы

15.05.2025

Вакцинация

20.04.2025

Первичный прием ветеринарного врача

Приемы

Расписание

Клиенты

Профиль

Рисунок 41 – Страница истории посещений питомца

На макете страницы истории посещений питомца имеются следующие элементы:

- 1 – поля с информацией о питомце;
- 2 – список прошедших посещений питомца, по нажатию на которые открывается страница приема.

Интерфейс страницы добавления расписания ветеринарного врача представлен на рисунке 42.

Расписание

Дни недели:

☐ Понедельник 1

☐ Вторник

☐ Среда

☐ Четверг

☐ Пятница

☐ Суббота

☐ Воскресенье

С: До: 2

Длительность приема (мин): 3

4

Слоты на неделю: 5

Понедельник, 20.04.2025
9:00 - 10:00

Понедельник, 20.04.2025
10:00 - 11:00

Приемы	Расписание	Клиенты	Профиль
--------	------------	---------	---------

Рисунок 42 – Страница добавления расписания врача

На макете страницы добавления расписания имеются следующие элементы:

- 1 – флажки для выбора рабочих дней недели;
- 2 – поля для указания времени работы;
- 3 – поле указания длительности приема в минутах;
- 4 – кнопка добавления расписания, по нажатию на которую в базу данных добавляются слоты для записи на неделю;
- 5 – список добавленных слотов для записи на прием на неделю.

После авторизации пользователя как администратор клиники, пользователю доступны дополнительные страницы: добавления и удаления ветеринарных врачей ветеринарной клиники, добавления и удаления услуг и цен ветеринарной клиники. Макет интерфейса страницы добавления и удаления врачей представлен на рисунке 43.

Макет интерфейса страницы «Ветеринарные врачи» представлен на рисунке 43. Он включает следующие элементы:

- Ветеринарные врачи** (Заголовок)
- Выбор врача:** (Текстовая метка)
- 1** (Пользовательский номер для выпадающего списка выбора врача)
- Удалить врача** (Кнопка)
- 2** (Пользовательский номер для кнопки удаления)
- Добавление врача:** (Текстовая метка)
- 3** (Пользовательский номер для поля ввода фамилии)
- Фамилия** (Поле ввода)
- 4** (Пользовательский номер для поля ввода имени)
- Имя** (Поле ввода)
- 5** (Пользовательский номер для поля ввода отчества)
- Отчество** (Поле ввода)
- 6** (Пользовательский номер для поля ввода номера телефона)
- Номер телефона** (Поле ввода)
- 7** (Пользовательский номер для поля ввода email)
- email** (Поле ввода)
- 8** (Пользовательский номер для поля ввода специализации)
- Специализация** (Поле ввода)
- 9** (Пользовательский номер для поля ввода услуг)
- Услуги** (Поле ввода)
- 10** (Пользовательский номер для кнопки добавления)
- Добавить врача** (Кнопка)

Рисунок 43 – Страница добавления и удаления врачей

На макете страницы добавления и удаления ветеринарных врачей имеются следующие элементы:

- 1 – поле с выпадающим списком для выбора врача;
- 2 – кнопка удаления ветеринарного врача, по нажатию на которую врач удаляется из базы данных;
- 3 – поле ввода фамилии врача;
- 4 – поле ввода имени врача;

- 5 – поле ввода отчества врача;
- 6 – поле ввода номера телефона врача;
- 7 – поле ввода email врача;
- 8 – поле ввода специализации врача;
- 9 – поле выбора услуг, предоставляемых врачом;
- 10 – кнопка добавления врача, по нажатию на которую врач добавляется в базу данных.

Макет интерфейса страницы добавления и удаления услуг представлен на рисунке 44.

The wireframe shows a page titled 'Услуги' (Services). It contains two main sections: 'Выбор услуги:' (Service selection) and 'Добавление услуги:' (Service addition). The 'Выбор услуги:' section has a dropdown menu (labeled 1) and a 'Удалить услугу' (Delete service) button (labeled 2). The 'Добавление услуги:' section has three input fields: 'Название' (Name) (labeled 3), 'Цена' (Price) (labeled 4), and 'Описание' (Description) (labeled 5). Below these fields is a 'Добавить услугу' (Add service) button (labeled 6).

Рисунок 44 – Страница добавления и удаления услуг

На макете страницы добавления и удаления услуг, предоставляемых ветеринарной клиникой, имеются следующие элементы:

- 1 – поле с выпадающим списком для выбора услуги;

2 – кнопка удаления услуги, по нажатию на которую услуга удаляется из базы данных;

3 – поле ввода названия услуги;

4 – поле ввода цены на услугу;

5 – поле ввода описания услуги;

6 – кнопка добавления услуги, по нажатию на которую услуга добавляется в базу данных.

3 Реализация программного обеспечения

3.1 Реализация базы данных

На основании концептуальной и логической модели для СУБД Microsoft SQL Server 2019 была разработана физическая модель базы данных [21]. Она представлена на рисунке 45.

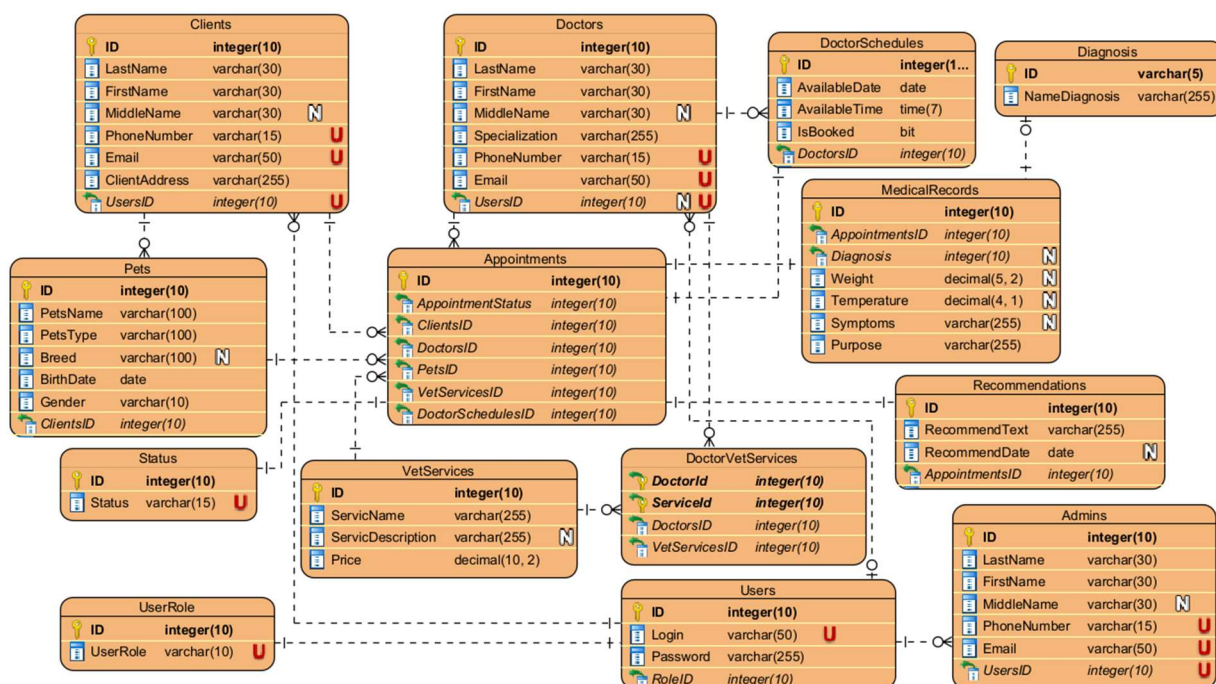


Рисунок 45 – Графическое представление физической модели базы данных

Описание атрибутов сущности «Клиент» представлено в таблице 12.

Таблица 12 – Описание атрибутов сущности «Клиент»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
LastName	Фамилия	varchar(30)	Нет	Нет	-
FirstName	Имя	varchar(30)	Нет	Нет	-
MiddleName	Отчество	varchar(30)	Да	Нет	-
PhonNumber	Номер телефона	varchar(15)	Нет	Да	-
Email	Почта	varchar(30)	Нет	Да	-
ClientAdress	Адрес	varchar(255)	Да	Нет	-
UsersID	Идентификатор пользователя	integer	Нет	Да	FK

Описание атрибутов сущности «Пользователь» представлено в таблице 13.

Таблица 13 – Описание атрибутов сущности «Пользователь»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
Login	Логин пользователя	varchar(30)	Нет	Да	-
Password	Пароль пользователя	varchar(30)	Нет	Нет	-
RoleID	Идентификатор роли пользователя	integer	Нет	Нет	FK

Описание атрибутов сущности «Врачи» представлено в таблице 14.

Таблица 14 – Описание атрибутов сущности «Врачи»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
LastName	Фамилия	varchar(30)	Нет	Нет	-
FirstName	Имя	varchar(30)	Нет	Нет	-
MiddleName	Отчество	varchar(30)	Да	Нет	-
Specialization	Специальность врача	varchar(255)	Нет	Нет	-
PhonNumber	Номер телефона	varchar(15)	Нет	Да	-
Email	Почта	varchar(30)	Нет	Да	-
UsersID	Идентификатор пользователя	integer	Нет	Да	FK

Описание атрибутов сущности «Администраторы» представлено в таблице 15.

Таблица 15 – Описание атрибутов сущности «Администраторы»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
LastName	Фамилия	varchar(30)	Нет	Нет	-
FirstName	Имя	varchar(30)	Нет	Нет	-
MiddleName	Отчество	varchar(30)	Да	Нет	-
PhonNumber	Номер телефона	varchar(15)	Нет	Да	-
Email	Почта	varchar(30)	Нет	Да	-
UsersID	Идентификатор пользователя	integer	Нет	Да	FK

Описание атрибутов сущности «Питомцы» представлено в таблице 16.

Таблица 16 – Описание атрибутов сущности «Питомцы»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
PetsName	Кличка питомца	varchar(100)	Нет	Нет	-
PetsType	Название животного	varchar(100)	Нет	Нет	-
Breed	Порода	varchar(100)	Да	Нет	-
BirthDate	Дата рождения	Date	Нет	Нет	-
Gender	Пол	varchar(10)	Нет	Нет	-
ClientID	Идентификатор клиента	integer	Нет	Нет	FK

Описание атрибутов сущности «Расписание врача» представлено в таблице 17.

Таблица 17 – Описание атрибутов сущности «Расписание врача»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
AvailableDate	Дата записи	date	Нет	Нет	-
AvailableTime	Время записи	time	Нет	Нет	-
IsBooked	Флаг брони	bit	Нет	Нет	-
DoctorsID	Идентификатор врача	integer	Нет	Нет	FK

Описание атрибутов сущности «Записи» представлено в таблице 18.

Таблица 18 – Описание атрибутов сущности «Записи»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
AppointmentStatus	Идентификатор статуса записи	integer	Нет	Нет	FK
ClientsID	Идентификатор клиента	integer	Нет	Нет	FK
DoctorsID	Идентификатор врача	integer	Нет	Нет	FK
PetsID	Идентификатор питомца	integer	Нет	Нет	FK
VetServicesID	Идентификатор услуги	integer	Нет	Нет	FK
DoctorSchedulesID	Идентификатор расписания	integer	Да	Нет	FK

Описание атрибутов сущности «Рекомендации» представлено в таблице 19.

Таблица 19 – Описание атрибутов сущности «Рекомендации»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
RecommendText	Текст рекомендации	varchar(255)	Нет	Нет	-
RecommendDate	Дата рекомендации	date	Нет	Нет	
AppointmentsID	Идентификатор записи	integer	Нет	Нет	FK

Описание атрибутов сущности «Медицинские карты» представлено в таблице 20.

Таблица 20 – Описание атрибутов сущности «Медицинские карты»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
AppointmentsID	Идентификатор записи	integer	Нет	Нет	FK
Diagnosis	Диагноз	varchar(5)	Нет	Нет	FK
Weight	Вес питомца	decimal(5, 2)	Нет	Нет	-
Temperature	Температура питомца	decimal(4, 1)	Нет	Нет	-
Symptoms	Симптомы	varchar(255)	Да	Нет	-
Purpose	Цель визита к врачу	varchar(255)	Нет	Нет	-

Описание атрибутов сущности «Услуги» представлено в таблице 21.

Таблица 21 – Описание атрибутов сущности «Услуги»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
ServicName	Название услуги	varchar(255)	Нет	Нет	-
ServicDescription	Описание услуги	varchar(255)	Да	Нет	-
Price	Цена	decimal(10, 2)	Нет	Нет	-

Описание атрибутов сущности «Статус» представлено в таблице 22.

Таблица 22 – Описание атрибутов сущности «Статус»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
Status	Статус записи	varchar(15)	Нет	Да	-

Описание атрибутов сущности «Роль пользователя» представлено в таблице 23.

Таблица 23 – Описание атрибутов сущности «Роль пользователя»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор	integer	Нет	Да	PK
UserRole	Роль пользователя	varchar(10)	Нет	Да	-

Описание атрибутов сущности «Диагнозы» представлено в таблице 24.

Таблица 24 – Описание атрибутов сущности «Диагнозы»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ID	Идентификатор диагноза	varchar(5)	Нет	Да	PK
NameDiagnosis	Название диагноза	varchar(255)	Нет	Да	-

Описание атрибутов сущности «Услуги врача» представлено в таблице 25.

Таблица 25 – Описание атрибутов сущности «Услуги врача»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
DoctorID	Идентификатор врача	integer	Нет	Нет	PK, FK
ServiceID	Идентификатор услуги	integer	Нет	Нет	PK, FK

На данной физической модели структура базы данных отображена с учетом спецификации типов СУБД Microsoft SQL Server 2019.

На основе физической модели БД был сгенерирован скрипт на языке SQL для создания базы данных. Его код представлен в приложении Б. Развертывание спроектированной базы данных было выполнено путем запуска данного скрипта в СУБД Microsoft SQL Server 2019.

3.2 Реализация серверной части приложения

Реализация серверной части приложения была произведена в среде программирования Microsoft Visual Studio 2022 на языке C# с применением

технологии Web API на платформе ASP.NET [22, 23]. Для взаимодействия с созданной базой данных была использована технология Entity Framework [24].

На основании используемой архитектуры MVC все классы серверной части делятся на два слоя: слой модели, предназначенный для взаимодействия с базой данных, и слой контроллеров, предназначенный для обработки HTTP-запросов к серверу.

Диаграмма классов слоя модели представлена на рисунке 46.

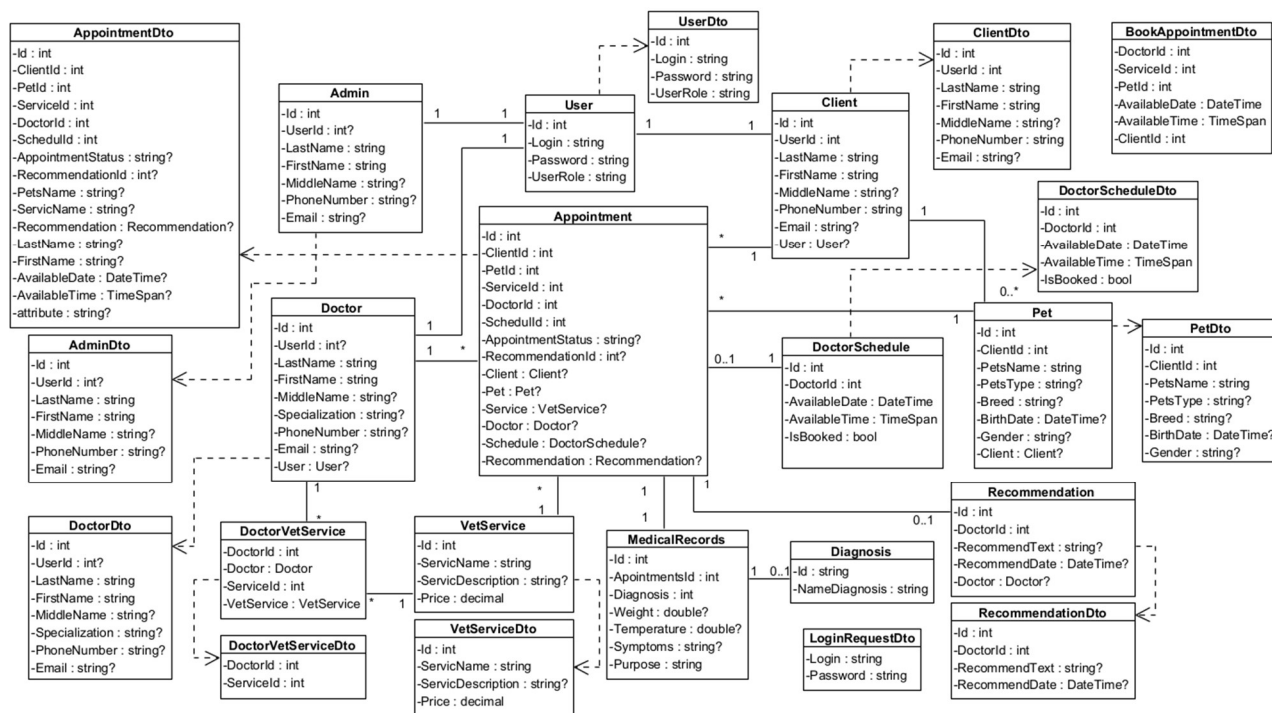


Рисунок 46 – Диаграмма классов слоя модели

Как можно увидеть из диаграммы, классы Admin, Appointment, Client, Diagnosis, Doctor, DoctorSchedule, DoctorVetService, MedicalRecords, Pet, Recommendation, User, VetService были созданы на основании одноименных таблиц базы данных и представляют сущности базы данных. Классы AdminDto, AppointmentDto, ClientDto, DoctorDto, DoctorScheduleDto, DoctorVetServiceDto, MedicalRecordsDto, PetDto, RecommendationDto, UserDto VetServiceDto нужны для безопасной передачи данных.

Диаграмма классов слоя контроллеров представлена на рисунке 47.

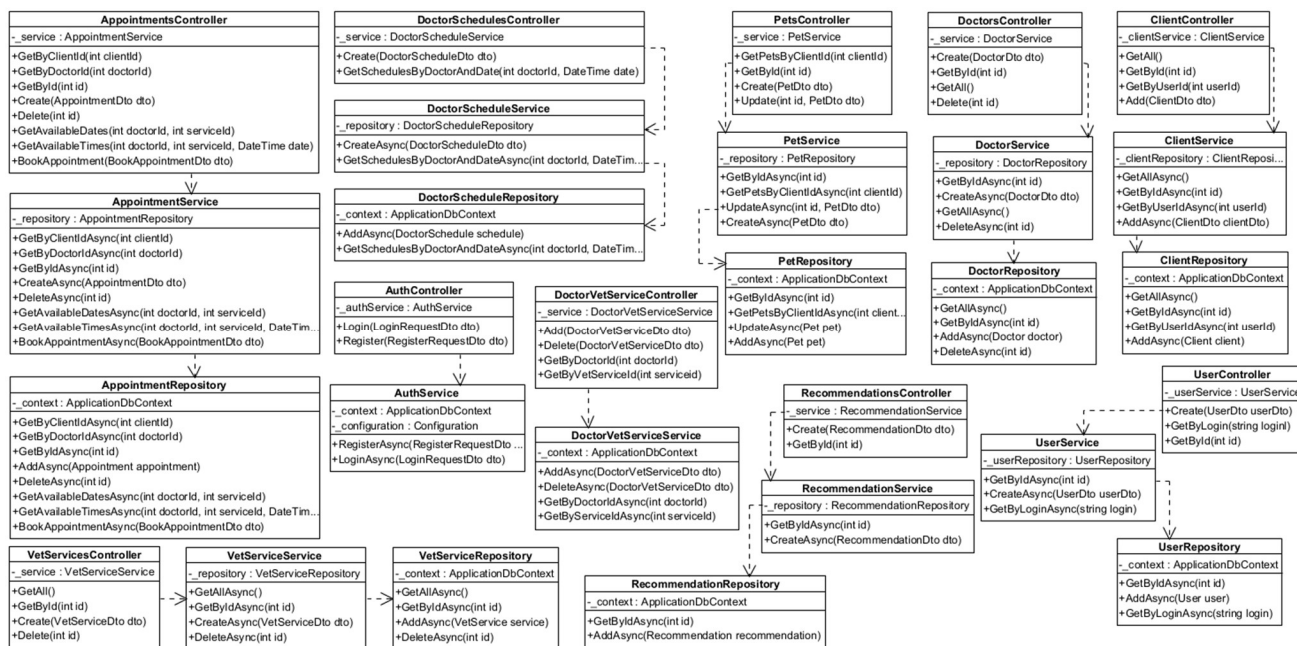


Рисунок 47 – Диаграмма классов слоя контроллеров

Как можно увидеть из диаграммы, слой контроллеров состоит из файлов Repositories, обеспечивающих доступ к данным, Services, содержащие бизнес-логику, и Controllers, отвечающие за маршрутизацию запросов. Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 26.

Таблица 26 – Описание классов слоя контроллеров

Класс	Назначение
AppointmentsController	Отвечает за обработку запросов, связанных с записями (например запись на прием, получение запланированных и прошедших приемов).
AppointmentsService	Управляет логикой обработки запросов, связанных с записями (например проверяет доступность, бронирует расписание)
AppointmentsRepository	Обращается к таблице записей
AuthController	Отвечает за обработку запросов на регистрацию и вход.
AuthService	Управляет логикой обработки запросов (проверяет данные, определяет роль).
ClientController	Отвечает за обработку запросов, связанных с клиентами (например добавление клиента, получение информации о клиенте).
ClientService	Управляет логикой обработки запросов, связанных с клиентами.
ClientRepository	Обращается к таблице клиентов.
DoctorSchedulesController	Отвечает за обработку запросов, связанных с расписанием (например добавление расписания, получение расписания).
DoctorSchedulesService	Управляет логикой обработки запросов, связанных с расписанием (например проверяет доступность, фильтрация)
DoctorSchedulesRepository	Обращается к таблице расписания

Продолжение таблицы 26

Класс	Назначение
DoctorsController	Отвечает за обработку запросов, связанных с врачами (например добавление врача).
DoctorsService	Управляет логикой обработки запросов, связанных с врачами.
DoctorsRepository	Обращается к таблице врачей.
DoctorVetServiceController	Отвечает за обработку запросов на услуги, оказываемые врачом.
DoctorVetServiceService	Управляет логикой обработки запросов на услуги, оказываемые врачом
MedicalRecordsController	Отвечает за обработку запросов, связанных с медицинскими картами (например добавление карты, получение).
MedicalRecordsService	Управляет логикой обработки запросов, связанных с медицинскими картами.
MedicalRecordsRepository	Обращается к таблице медицинских карт.
PetsController	Отвечает за обработку запросов, связанных с питомцами (например добавление питомца, получение питомцев клиента).
PetsService	Управляет логикой обработки запросов, связанных с питомцами.
PetsRepository	Обращается к таблице питомцев.
RecommendationsController	Отвечает за обработку запросов, связанных с рекомендациями врача (например добавление рекомендации).
RecommendationsService	Управляет логикой обработки запросов, связанных с рекомендациями врача.
RecommendationsRepository	Обращается к таблице рекомендаций врача.
UserController	Отвечает за обработку запросов, связанных с пользователями (например получение пользователя по логину).
UserService	Управляет логикой обработки запросов, связанных с пользователями.
UserRepository	Обращается к таблице пользователей.
VetServicesController	Отвечает за обработку запросов, связанных с услугами (например добавление, получение всех услуг).
VetServicesService	Управляет логикой обработки запросов, связанных с услугами.
VetServicesRepository	Обращается к таблице услуг.

Листинг серверной части представлен в приложении В.

3.3 Реализация мобильного приложения

На данном этапе было реализовано мобильное приложение на языке C# в среде Visual Studio 2022. Исходный код представлен в приложении Г. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 48.

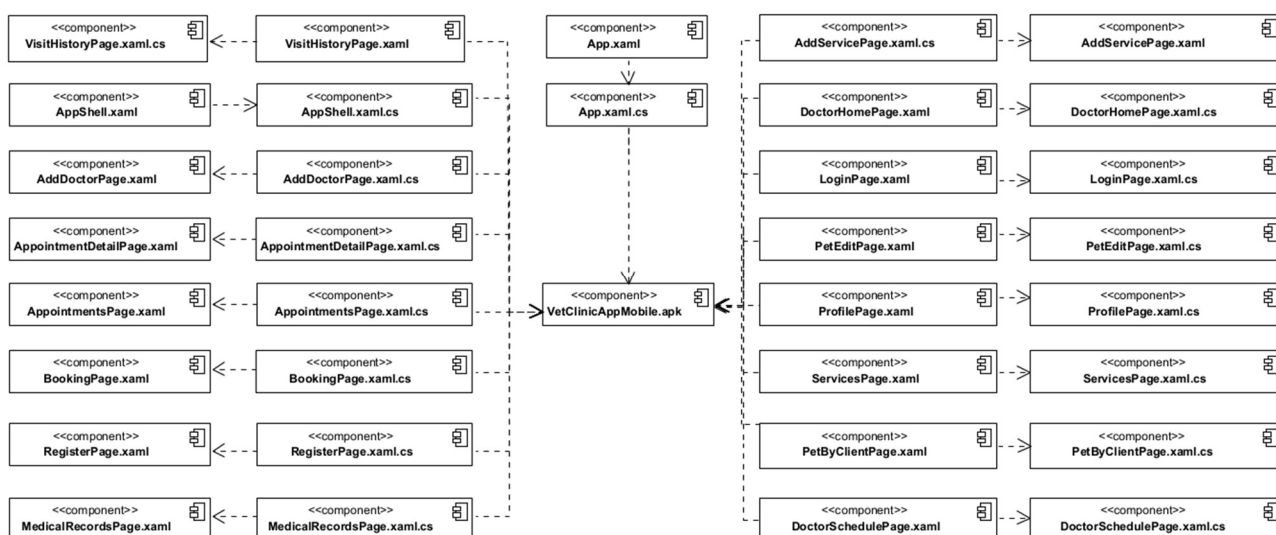


Рисунок 48 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 27.

Таблица 27 – Описание компонентов мобильного приложения

Компонент	Назначение
App.xaml	XAML-файл описания подключаемых ресурсов к приложению
App.xaml.cs	Начальная точка загрузки и закрытия приложения
AppShell.xaml	XAML-файл описания структуры интерфейса приложения
AppShell.xaml.cs	Регистрирует страницы приложения, не вошедшие в основную структуру
VisitHistory.xaml	XAML-файл описания интерфейса страницы истории посещений питомца клиники
VisitHistory.xaml.cs	Управляет логикой и данными страницы истории посещений питомца клиники
AddDoctorPage.xaml	XAML-файл описания интерфейса страницы добавления нового ветеринарного врача админом
AddDoctorPage.xaml.cs	Управляет логикой и данными страницы добавления нового ветеринарного врача админом
AppointmentDetailPage.xaml	XAML-файл описания интерфейса страницы деталей приема и рекомендаций врача
AppointmentDetailPage.xaml.cs	Управляет логикой и данными страницы деталей приема и рекомендаций врача
AppointmentsPage.xaml	XAML-файл описания интерфейса страницы запланированных и прошедших приемов клиента
AppointmentsPage.xaml.cs	Управляет логикой и данными страницы запланированных и прошедших приемов клиента
BookingPage.xaml	XAML-файл описания интерфейса страницы записи к ветеринарному врачу
BookingPage.xaml.cs	Управляет логикой и данными страницы записи к ветеринарному врачу
RegisterPage.xaml	XAML-файл описания интерфейса страницы регистрации нового пользователя
RegisterPage.xaml.cs	Управляет логикой и данными страницы регистрации нового пользователя

Продолжение таблицы 27

Компонент	Назначение
MedicalRecords.xaml	ХАМЛ-файл описания интерфейса страницы добавления медицинской карточки питомца
MedicalRecords.xaml.cs	Управляет логикой и данными страницы добавления медицинской карточки питомца
AddServicePage.xaml	ХАМЛ-файл описания интерфейса страницы добавления новых услуг клиники админом
AddServicePage.xaml.cs	Управляет логикой и данными страницы добавления новых услуг клиники админом
DoctorHomePage.xaml	ХАМЛ-файл описания интерфейса страницы приемов врача
DoctorHomePage.xaml.cs	Управляет логикой и данными страницы приемов врача
LoginPage.xaml	ХАМЛ-файл описания интерфейса страницы входа в систему
LoginPage.xaml.cs	Управляет логикой и данными страницы входа в систему
PetEditPage.xaml	ХАМЛ-файл описания интерфейса страницы добавления питомца
PetEditPage.xaml.cs	Управляет логикой и данными страницы добавления питомца
ProfilePage.xaml	ХАМЛ-файл описания интерфейса страницы профиля пользователя
ProfilePage.xaml.cs	Управляет логикой и данными страницы профиля пользователя
ServicesPage.xaml	ХАМЛ-файл описания интерфейса страницы услуг и их цен
ServicesPage.xaml.cs	Управляет логикой и данными страницы услуг и их цен
DoctorSchedulePage.xaml	ХАМЛ-файл описания интерфейса страницы добавления расписания врачом
DoctorSchedulePage.xaml.cs	Управляет логикой и данными страницы добавления расписания врачом
PetByClient.xaml	ХАМЛ-файл описания интерфейса страницы поиска питомцев по клиентам
PetByClient.xaml.cs	Управляет логикой и данными страницы поиска питомцев по клиентам
VetClinicAppMobile.apk	Исполняемый файл мобильного приложения

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 49.

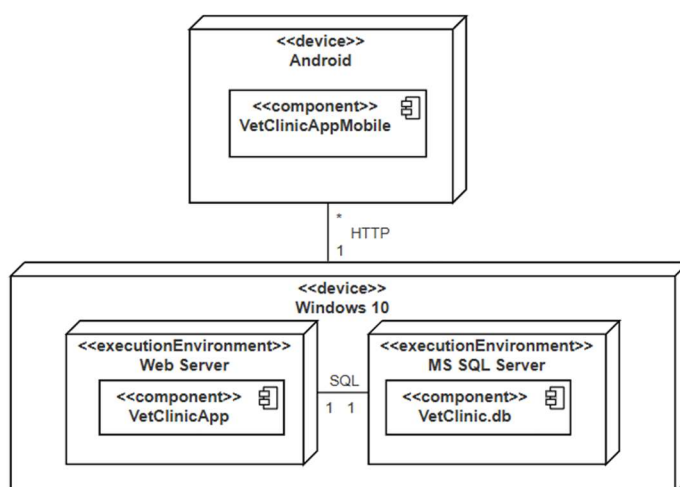


Рисунок 49 – Диаграмма развертывания программного обеспечения

Из данной диаграммы можно увидеть, что программное обеспечение состоит из трех уровней: мобильное приложение, работающее под управлением ОС Android, серверная часть и СУБД, работающие под управлением ОС Windows 10.

3.4 Тестирование программного обеспечения

3.4.1 Тестирование серверной части приложения

3.4.1.1 Функциональное тестирование

На данном этапе было выполнено функциональное тестирование в целях проверки правильности работы серверной части приложения. Результаты функционального тестирования серверной части приложения представлены в таблице 28.

Таблица 28 – Результаты функционального тестирования серверной части

№	Описание теста	Структура запроса	Результат
1	Авторизация клиента	Метод: POST URL: api/Auth/login Body: {"login": "client1petr", "password": "pasS12"}	Тест пройден. Клиент авторизован (рисунок Д.1).
2	Авторизация ветеринарного врача	Метод: POST URL: api/Auth/login Body: {"login": "terapevt1", "password": "parol234"}	Тест пройден. Ветеринарный врач авторизован (рисунок Д.2).
3	Получение услуг	Метод: GET URL: api/VetServices	Тест пройден. Список услуг получен (рисунок Д.3).
4	Получение записей клиента	Метод: GET URL: /api/Appointments/client/1	Тест пройден. Список записей получен (рисунок Д.4).
5	Получение времени для записи к врачу на выбранную дату	Метод: GET URL: api/DoctorSchedules/doctor/1/date/2025-05-21T00%3A00%3A00	Тест пройден. Доступное время для записи получено (рисунок Д.5).
6	Получение информации о клиенте	Метод: GET URL: /api/Client/1	Тест пройден. Информация о клиенте получена (рисунок Д.6).
7	Регистрация клиента	Метод: POST URL: api/Auth/register Body: {"login": "clienttest", "password": "pass_234", "lastName": "Ежов", "firstName": "Лев", "middleName": "Иванович", "email": "clienttest@gmail.com", "phoneNumber": "86756893345"}	Тест пройден. Новый клиент зарегистрирован в системе (рисунок Д.7).

Продолжение таблицы 28

№	Описание теста	Структура запроса	Результат
8	Получение всех питомцев клиента	Метод: GET URL: api/Pets/client/1	Тест пройден. Питомцы клиента получены (рисунок Д.8).
9	Получение услуг, предоставляемых врачом	Метод: GET URL: api/DoctorVetService/doctor/1	Тест пройден. Услуги, предоставляемые врачом получены (рисунок Д.9).
10	Запись к ветеринарному врачу	Метод: POST URL: /api/Appointments/book Body: {"doctorId": 1, "serviceId": 2, "petId": 1, "availableDate": "2025-05-19T00:00:00", "availableTime": "12:00:00", "clientId": 1}	Тест пройден. Запись к врачу успешно добавлена (рисунок Д.10).
11	Редактирование питомца	Метод: PUT URL: api/Pets/3 Body: {"id": 3, "clientId": 1, "petsName": "Снежка", "petsType": "Кошка", "breed": "Шотландская вислоухая", "birthDate": "2015-07-18T00:00:00", "gender": "Женский"}	Тест пройден. Питомец успешно изменен (рисунок Д.11).
12	Добавление питомца	Метод: POST URL: api/Pets Body: {"clientId": 1, "petsName": "Пушок", "petsType": "Собака", "breed": "Йоркшерский терьер", "birthDate": "2022-05-13T19:16:48.894Z", "gender": "Мужской"}	Тест пройден. Новый питомец успешно добавлен в систему (рисунок Д.12).
13	Получение записей ветеринарного врача	Метод: GET URL: /api/Appointments/doctor/1	Тест пройден. Получен список записей врача (рисунок Д.13).
14	Добавление слота для записи к врачу	Метод: POST URL: api/DoctorSchedules Body: {"doctorId": 1, "availableDate": "2025-06-05T19:21:28.154Z", "availableTime": "12:00:00", "isBooked": false}	Тест пройден. Новый слот для записи к врачу добавлен (рисунок Д.14).
15	Добавление ветеринарного врача	Метод: POST URL: api/Doctor Body: {"lastName": "Львов", "firstName": "Кирилл", "middleName": "Петрович", "specialization": "Врач-кардиолог", "phoneNumber": "895678564826", "email": "lvovkir@mail.com"}	Тест пройден. Новый ветеринарный врач добавлен в базу данных (рисунок Д.15).

Окончание таблицы 28

№	Описание теста	Структура запроса	Результат
16	Добавление медицинской карточки питомца	Метод: POST URL: api/MedicalRecords Body: {"id": 0, "apointmentsId": 7, "diagnosis": "A11", "weight": 4.0, "temperature": 38.4, "symptoms": "Вялость, плохой аппетит", "purpose": "Осмотр врача"}	Тест пройден. Новая медицинская карточка питомца добавлена в базу данных (рисунок Д.16).
17	Добавление рекомендации	Метод: POST URL: api/Recommendations Body: {"id": 7, "doctorId": 1, "recommendText": "Прием препарата Ципровет по 1 таблетке 1 раз в день", "recommendDate": "2025-05-19T19:29:39.882Z"}	Тест пройден. Рекомендация добавлена в базу данных (рисунок Д.17).
18	Удаление ветеринарного врача	Метод: DELETE URL: api/Doctors/8	Тест пройден. Ветеринарный врач удален из базы данных (рисунок Д.18).

Все тесты тестирования были пройдены. Значит можно говорить о правильности работы серверной части приложения.

3.4.1.2 Модульное тестирование

В дополнении к функциональному тестированию было проведено модульное тестирование серверной части для запроса получения расписания врача и доступного для записи времени. Оно выполнялось на языке программирования C# в среде разработки Visual Studio 2022. Исходный код тестов приведен в приложении Е, а его результаты – на рисунке 50.

Тестирование	Длительн...	Признаки	Сообщение
▲ ✓ VetClinic.Test (7)	12 мс		
▲ ✓ VetClinicTest (7)	12 мс		
▲ ✓ VetClinicTests (7)	12 мс		
✓ BookAppointment_ShouldFail_I...	5 мс		
✓ BookAppointment_ShouldIncre...	5 мс		
✓ BookAppointment_ShouldSucc...	< 1 мс		
✓ GetAllServices_ShouldReturnSer...	< 1 мс		
✓ GetAppointmentsByClientId_Sh...	2 мс		
✓ GetAppointmentsByClientId_Sh...	< 1 мс		
✓ GetPetsByClientId_ShouldRetur...	< 1 мс		

Рисунок 50 – Результаты модульного тестирования

Все тесты модульного тестирования прошли успешно. Следовательно, можно говорить о корректной работе данных функций.

3.4.2 Тестирование клиентской части приложения

Также было произведено функциональное тестирование мобильного приложения, результаты которого приведены в таблице 29.

Таблица 29 – Результаты функционального тестирования мобильного приложения

Вариант использования	Параметры теста	Результат
Авторизоваться	Авторизация клиента с корректными данными. Логин: clienttest Пароль: password15	Тест пройден. Авторизация выполнена (рисунок Ж.1).
	Авторизация клиента с некорректными данными. Логин: petr Пароль: petr12	Тест пройден. Показано сообщение с ошибкой (рисунок Ж.2).
	Авторизация врача с корректными данными. Логин: terapevt1 Пароль: password15	Тест пройден. Авторизация выполнена (рисунок Ж.3).
	Авторизация администратора с корректными данными. Логин: admin2 Пароль: pass_wor8	Тест пройден. Авторизация выполнена (рисунок Ж.4).
Записаться к ветеринарному врачу	Выбор услуги, ветеринарного врача, питомца, даты и времени и записи на прием.	Тест пройден. Клиент успешно записался на прием к ветеринарному врачу (рисунок Ж.5).
Посмотреть историю посещений	Переход на страницу приемов	Тест пройден. Запланированные приемы и история прошедших посещений успешно отображена (рисунок Ж.6).
Посмотреть услуги и цены	Переход на страницу услуг и цен.	Тест пройден. Услуги и цены успешно отображены (рисунок Ж.7).
Добавить питомца	Добавление клиентом нового питомца.	Тест пройден. Новый питомец успешно добавлен (рисунок Ж.8).
Добавить медицинскую карточку питомца	Заполнение врачом цели посещения, симптомов, диагноза, рекомендаций и статуса приема, и добавление медицинской карточки в систему.	Тест пройден. Медицинская карточка успешно добавлена (рисунок Ж.9).
Отменить запись к ветеринару	Отмена клиентом запланированной записи на прием.	Тест пройден. Запись успешно отменена и удалена из системы (рисунок Ж.10).
Добавить расписание	Добавление врачом расписания и новых слотов для записи.	Тест пройден. Новые слоты для записи на прием успешно добавлены (рисунок Ж.11).

Продолжение таблицы 29

Вариант использования	Параметры теста	Результат
Посмотреть рекомендации	Переход клиента на страницу рекомендаций после прошедшего приема.	Тест выполнен. Рекомендации после выбранного приема успешно отображены (рисунок Ж.12).
Добавить ветеринарного врача	Добавление администратором нового ветеринарного врача.	Тест выполнен. Ветеринарный врач добавлен в систему (рисунок Ж.13).
Добавить услуги и цены	Добавление администратором новой услуги, ее описания и цены.	Тест выполнен. Услуга успешно добавлена (рисунок Ж.14).

Все тесты были выполнены успешно. Следовательно, можно говорить о корректной работе мобильного приложения.

3.4.3 Метрики кода

В разработке программного обеспечения используются различные методы количественной оценки, которые позволяют проанализировать различные аспекты кода программы. Среди множества существующих подходов можно выделить две основные категории:

- 1 – Метрики объема кода;
- 2 – Метрики сложности кода.

Первый подход учитывает показатели подсчета строк исходного кода (Source Lines of Code – SLOC). Данный подход позволяет получить базовое представление о масштабах проекта и трудоемкости разработки.

Второй подход предполагает использование метрик, предоставляющих более содержательную информацию о коде и основанных на анализе структуры программы. Классическим примером подхода являются метрики Холстеда.

Рассмотрим 2 метода класса BookingPage.xaml.cs: обработчик выбора даты для записи (OnDateSelected) и обработчик нажатия кнопки записи (OnBookClicked). Коды данных методов приведены в листинге 1.

Листинг 1 – Коды методов класса BookingPage

```
// Обработчик выбора даты
private async void OnDateSelected(object sender, EventArgs e)
{
    // Проверка выбранных значений
    if (DatePicker.SelectedItem is not string dateString ||
        ServicePicker.SelectedItem is not VetServiceDto service ||
        DoctorPicker.SelectedItem is not DoctorDto doctor)
```

```

{
    TimePicker.ItemsSource = null;
    return;
}
// Парсинг даты
if (!DateTime.TryParse(dateString, out var selectedDate))
{
    TimePicker.ItemsSource = null;
    return;
}
// Получение доступного времени
var times = await _api.GetAvailableTimesAsync(doctor.Id,
service.Id, selectedDate);
// Проверка наличия доступного времени
if (times == null || times.Count == 0)
{
    TimePicker.ItemsSource = new List<string> {"Нет доступного
времени" };
    TimePicker.SelectedItem = "Нет доступного времени";
    return;
}
// Форматирование времени
var timeStrings = times.Select(t =>
t.ToString(@"hh\:mm")).ToList();
TimePicker.ItemsSource = timeStrings;
TimePicker.SelectedItem = null;
}
// Обработчик нажатия кнопки записи
private async void OnBookClicked(object sender, EventArgs e)
{
    // Проверка заполнения всех полей
    if (ServicePicker.SelectedItem is not VetServiceDto service ||
DoctorPicker.SelectedItem is not DoctorDto doctor ||
DatePicker.SelectedItem is not string dateString ||
TimePicker.SelectedItem is not string timeString ||
PetPicker.SelectedItem is not PetDto pet ||
dateString == "Нет доступных дат" || timeString == "Нет доступного
времени")
    {
        await DisplayAlert("Ошибка", "Пожалуйста, заполните все
поля", "OK");
        return;
    }
    // Парсинг даты и времени
    if (!DateTime.TryParse(dateString, out var date) ||
!TimeSpan.TryParse(timeString, out var time))
    {
        await DisplayAlert("Ошибка", "Некорректная дата или
время", "OK");
        return;
    }
    // Создание DTO для записи
    var dto = new BookAppointmentDto

```

```

{
    ServiceId = service.Id,
    DoctorId = doctor.Id,
    ClientId = Preferences.Get("ClientId", 0),
    AvailableDate = date,
    AvailableTime = time,
    PetId = pet.Id,
};
// Отправка запроса на сервер
bool success = await _api.BookAppointmentAsync(dto);
// Обработка результата
if (success)
{
    await DisplayAlert("Успех", "Вы успешно записались на
прием", "OK");
    await Shell.Current.GoToAsync("//AppointmentsPage");
}
else
{
    await DisplayAlert("Ошибка", "Не удалось записаться.
Попробуйте позже.", "OK");
}
}

```

Данные методы находятся в клиентской части приложения. Эти методы представляют собой обработчик выбора даты после выбора ветеринарного врача и обработчик нажатия кнопки записи. Функция `OnDateSelected` вызывается при выборе даты в интерфейсе пользователя и отвечает за проверку заполнения необходимых полей, получение доступного времени для записи и обновление доступного времени для записи в интерфейсе. Функция `OnBookClicked` вызывается при нажатии кнопки «Записаться» и отвечает за проверку заполнения всех полей формы и создание объекта записи. Затем метод выполняет HTTP-запрос к серверной части приложения для записи клиента к ветеринарному врачу. Если данный запрос успешный, то возвращается сообщение об успешной записи, иначе возвращается сообщение об ошибке.

Для данных методов будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

3.4.3.1 Метрики размера программы

Данная группа метрик используется для измерения объема программы с помощью подсчета количества строк в тексте исходного кода. При этом часто

выделяют методику подсчета физических строк кода и строк комментариев. Вычислим метрики размера программы для кода, приведенного в листинге 1 (таблица 30).

Таблица 30 – Метрики размера программы

Метрика	Количество
Количество строк кода	73
Количество строк комментариев	12

Теперь вычислим процент комментариев. Он равен $(12/73)*100\%=16\%$. Это означает, что количество комментариев в коде достаточно для его хорошего понимания, и при этом не слишком велико и не переполняет программу.

3.4.3.2 Метрики Холстеда

Метрика Холстеда рассчитывается на основе анализа числа строк и различных элементов синтаксиса исходного кода программы, а именно считаются такие характеристики, как число уникальных операторов, число уникальных операндов и общее число операторов [25]. Словарь операторов приведен в таблице 31, словарь операндов в таблице 32.

Таблица 31 – Словарь операторов

Оператор	Число вхождений
=	15
.	30
If	5
is not	6
var	6
!	2
	8
return	5
await	4
==	2
New	2
=>	1
!=	1
out	3
Count	1
null	3
;	28
[]	2
,	8
}	10
()	18
List	1
Shell	1
n1 = 23	N1 = 163

Таблица 32 – Словарь операндов

Операнд	Число вхождений
DatePicker.SelectedItem	3
TimePicker.ItemsSource	5
ServicePicker.SelectedItem	2
DoctorPicker.SelectedItem	2
PetPicker.SelectedItem	1
dateString	4
service	4
doctor	3
selectedDate	2
times	2
timeStrings	2
date	2
time	2
pet	2
dto	2
success	2
_api	2
GetAvailableTimesAsync	1
BookAppointmentAsync	1
DisplayAlert	3
Shell.Current.GoToAsync	1
"Нет доступного времени"	2
"Ошибка"	2
"Пожалуйста, заполните все поля"	1
"Некорректная дата или время"	1
"Успех"	1
"Вы успешно записались на прием"	1
"Не удалось записаться. Попробуйте позже "	1
"ОК"	3
"//AppointmentsPage"	1
DateTime.TryParse	1
TimeSpan.TryParse	1
Preferences.Get	1
BookAppointmentDto	1
VetServiceDto	2
DoctorDto	2
PetDto	1
t	1
@ "hh:mm"	1
0	1
n2 = 40	N2 = 73

Из вышеописанных таблиц получаем следующие данные: число уникальных операторов программы $n_1 = 23$, число уникальных операндов программы $n_2 = 40$, общее число операторов в программе $N_1 = 163$, общее число операндов в программе $N_2 = 73$.

Вычислим некоторые из существующих характеристик на основании полученных данных:

– длина программы (N):

$$N = N_1 + N_2 = 236;$$

- словарь программы (n):

$$n = n_1 + n_2 = 63;$$

- теоретический объем программы (V):

$$V = N \cdot \log_2 N \approx 1860.3;$$

- сложность программы (D):

$$D = \left(\frac{n_1}{2}\right) \cdot \left(\frac{N_2}{n_2}\right) \approx 20.9;$$

- оценка интеллектуальных усилий (E):

$$E = D \cdot V = 38880,3.$$

Вычисленные характеристики показывают, что программный код имеет достаточно высокий уровень качества и удобства для дальнейшего усовершенствования поддержки.

Заключение

В ходе выполнения выпускной квалификационной работы были созданы программные средства поддержки процессов функционирования ветеринарной клиники.

На этапе анализа предметной области и требований был выполнен анализ программного обеспечения аналогичного назначения, в результате которого были выявлены их основные достоинства и недостатки, произведен анализ целевой аудитории и определены функциональные и нефункциональные требования.

Также было осуществлено планирование разработки и оценка бюджета. По результатам планирования было определено, что общая длительность проекта составляет 51 день, а предполагаемые затраты будут составлять 206450 рублей. Таким образом можно сделать вывод об экономической целесообразности проекта.

На этапе проектирования была составлена концептуальная и логическая модель базы данных, а также разработана архитектура мобильного приложения. База данных была создана с использованием языка программирования SQL и развёрнута под управлением СУБД Microsoft SQL Server 2019. Серверная часть приложения была разработана на языке C# с использованием среды программирования Microsoft Visual Studio 2022. Клиентская часть была разработана на языке C# с использованием среды программирования Microsoft Visual Studio 2022.

На этапе реализации было выполнено физическое проектирование базы данных в СУБД Microsoft SQL Server 2019 и реализован весь спроектированный функционал. Было выполнено тестирование серверной и клиентской частей разрабатываемого приложения. Оно показало корректность его работы. Также рассчитаны метрики программного кода для оценки его качества.

В дальнейшем планируется опубликовать разработанное приложение в российских магазинах мобильных приложений для системы Android, таких как

RuStore, а также создать версию мобильного приложения для операционной системы iOS.

Таким образом, в ходе выполнения выпускной квалификационной работы были успешно выполнены все поставленные задачи.

Список использованных источников

1. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 17.04.2025).
2. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632–р. [Электронный ресурс]. URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 17.04.2025).
3. Официальный сайт ветеринарной клиники «Доктор Зоо» [Электронный ресурс]. URL: <https://doctorzoopnz.ru> (дата обращения 18.05.2025)
4. Официальный сайт ветеринарной клиники «Ветеринарная клиника врачей Сумбаевых» [Электронный ресурс]. URL: <https://vetpnz.ru> (дата обращения 18.04.2025)
5. Официальный сайт ветеринарного центра «Счастливый слон» [Электронный ресурс]. URL: <https://zoocenter58.ru> (дата обращения 18.04.2025)
6. Центр изучения питания и благополучия животных [Электронный ресурс]. URL: <https://petcarecenter.ru/about#info-block> (дата обращения 20.04.2025)
7. Global Smartphone Sales Share by Operating System // [Электронный ресурс]. URL: <https://www.counterpointresearch.com/insights/global-smartphone-os-market-share/> (дата обращения: 21.04.2025).
8. ОС «Аврора» [Электронный ресурс]. URL: <https://digital.gov.ru/activity/razvitie-it-otrasli/os-avrora> (дата обращения: 25.04.2025).
9. Статистика версий Android. [Электронный ресурс]. URL: https://4pda.to/2023/01/19/408761/vsyo_grustno_google_obnarodovala_statistiku_versions_android/ (дата обращения: 21.04.2025).

10. Mobile & Tablet Android Version Market Share Worldwide // Statcounter [Электронный ресурс]. URL: <https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide/#monthly-202403-202503> (дата обращения 21.04.2025)
11. Крачтен, Ф. Введение в Rational Unified Process, 2-е издание / Ф. Крачтен // М.: Вильямс, 2002. – 240 с.
12. Документация по ASP.NET. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/aspnet/core/> (дата обращения: 24.04.2025).
13. Беркун, С. Искусство управления IT-проектами / С. Беркун // СПб.: Питер, 2007. – 400 с.: ил.
14. Разу, М.Л. Управление проектом. Основы проектного управления: учебник, 4-е издание / М.Л. Разу, Т.М. Бронникова, А.М. Лялин // М.: КНОРУС, 2015. – 756 с.
15. Документация по Project. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/project/> (дата обращения: 11.03.2025).
16. Унифицированный процесс Rational (RUP) // Интернет–технологии.ру [Электронный ресурс]. – URL: <https://www.internet-technologies.ru/articles/newbie/unificirovannyu-process-rational-rup.html> (дата обращения: 07.05.2025).
17. Документация по Microsoft SQL. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/sql/> (дата обращения: 07.05.2025).
18. Основы Linq. [Электронный ресурс]. URL: <https://metanit.com/sharp/tutorial/15.1.php> (дата обращения: 02.05.2025).
19. Логическая модель базы данных. // Studopedia [Электронный ресурс]. URL: https://studopedia.ru/6_1895_logicheskaya-model-bazi-dannih.html (дата обращения 06.05.2025).
20. Шаблон Model-View-Controller. [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/aspnet/core/mvc> (дата обращения: 07.05.2025).

21. Физическая модель базы данных. // Studopedia [Электронный ресурс]. URL: https://studopedia.ru/6_1898_fizicheskaya-model-bazi-dannih.html (дата обращения 07.05.2025).
22. Введение в Web API // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/mvc/12.1.php> (дата обращения: 25.05.2025).
23. Андерсон Р., Ларкин К., Уоссон М. Создание веб-API с помощью ASP.NET Core и Visula Studio для Windows // Microsoft [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-3.1&tabs=visual-studio> (дата обращения: 25.05.2025).
24. Руководство по Entity Framework // Metanit.com [Электронный ресурс]. URL: <https://metanit.com/sharp/entityframework/2.4.php> (дата обращения: 26.05.2025).
25. Полякова, Э.Р. Исследование программного обеспечения встраиваемой вычислительной системы метриками Холстеда / Э.Р. Полякова, А.А. Поляков, А.Л. Шустов, Д.Г. Придачкин // Cloud of Science. – 2020. – №4. – С. 950 – 959.

Приложение А
(справочное)
Глоссарий

Android – мобильная операционная система для смартфонов, планшетов и других мобильных устройств, разрабатываемая и выпускаемая американской компанией Google и используемая на подавляющем большинстве современных мобильных устройств.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

RUP (Rational Unified Process) – методология разработки программного обеспечения, в основе которой лежит итеративная модель разработки.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания и модификации данных в реляционной базе данных, работающей под управлением соответствующей системой управления базами данных, а также для управления этими данными.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, основанный на технологии «клиент–сервер».

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес–процессов, системного проектирования и отображения организационных структур.

.NET MAUI (.NET Multi–Platform App UI) – это кросс–платформенная платформа для создания собственных мобильных и классических приложений с помощью C# и XAML.

Entity Framework – представляет специальную объектно–ориентированную технологию на базе фреймворка .NET для работы с данными, обладает высоким уровнем абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

Приложение Б
(обязательное)

Исходный код скрипта создания базы данных

```

CREATE TABLE Users (
    Id INT PRIMARY KEY IDENTITY(1,1),
    Login VARCHAR(25) UNIQUE NOT NULL,
    PasswordHash VARCHAR(255) NOT NULL,
    UserRole VARCHAR(10) NOT NULL -- Client, Doctor, Admin
);
CREATE TABLE Clients (
    Id INT PRIMARY KEY IDENTITY(1,1),
    UserId INT NOT NULL,
    LastName VARCHAR(30) NOT NULL,
    FirstName VARCHAR(30) NOT NULL,
    MiddleName VARCHAR(30),
    PhoneNumber VARCHAR(15) NOT NULL,
    Email VARCHAR(30),
    ClientAddress VARCHAR(255),
    FOREIGN KEY (UserId) REFERENCES Users(Id)
);
CREATE TABLE Doctors (
    Id INT PRIMARY KEY IDENTITY(1,1),
    UserId INT NOT NULL,
    LastName VARCHAR(30) NOT NULL,
    FirstName VARCHAR(30) NOT NULL,
    MiddleName VARCHAR(30),
    Specialization VARCHAR(255),
    PhoneNumber VARCHAR(15),
    Email VARCHAR(50),
    FOREIGN KEY (UserId) REFERENCES Users(Id)
);
CREATE TABLE Admins (
    Id INT PRIMARY KEY IDENTITY(1,1),
    UserId INT NOT NULL,
    LastName VARCHAR(30) NOT NULL,
    FirstName VARCHAR(30) NOT NULL,
    MiddleName VARCHAR(30),
    PhoneNumber VARCHAR(15),
    Email VARCHAR(50),
    FOREIGN KEY (UserId) REFERENCES Users(Id)
);
CREATE TABLE Pets (
    Id INT PRIMARY KEY IDENTITY(1,1),
    ClientId INT NOT NULL,
    PetsName VARCHAR(100) NOT NULL,
    PetsType VARCHAR(100),
    Breed VARCHAR(100),
    BirthDate DATE,
    Gender VARCHAR(10),
    FOREIGN KEY (ClientId) REFERENCES Clients(Id)
);
CREATE TABLE VetServices (
    Id INT PRIMARY KEY IDENTITY(1,1),
    ServName VARCHAR(255) NOT NULL,
    ServDescription VARCHAR,
    Price DECIMAL(10, 2) NOT NULL

```

```

);
CREATE TABLE DoctorSchedules (
    Id INT PRIMARY KEY IDENTITY(1,1),
    DoctorId INT NOT NULL,
    AvailableDate DATE NOT NULL,
    AvailableTime TIME NOT NULL,
    IsBooked BIT DEFAULT 0,
    FOREIGN KEY (DoctorId) REFERENCES Doctors(Id)
);
CREATE TABLE Appointments (
    Id INT PRIMARY KEY IDENTITY(1,1),
    ClientId INT NOT NULL,
    PetId INT NOT NULL,
    ServiceId INT NOT NULL,
    DoctorId INT NOT NULL,
    SchedulId INT NOT NULL,
    AppointmentDate DATE NOT NULL,
    AppointmentTime TIME NOT NULL,
    AppointmentStatus VARCHAR(50),
    FOREIGN KEY (ClientId) REFERENCES Clients(Id),
    FOREIGN KEY (PetId) REFERENCES Pets(Id),
    FOREIGN KEY (ServiceId) REFERENCES VetServices(Id),
    FOREIGN KEY (DoctorId) REFERENCES Doctors(Id),
    FOREIGN KEY (SchedulId) REFERENCES DoctorSchedules(Id)
);
CREATE TABLE Recommendations (
    Id INT PRIMARY KEY IDENTITY(1,1),
    AppointmentId INT NOT NULL,
    DoctorId INT NOT NULL,
    RecommendText VARCHAR,
    RecommendDate DATE,
    FOREIGN KEY (AppointmentId) REFERENCES Appointments(Id)
);
CREATE TABLE DoctorVetServices (
    DoctorId INT NOT NULL,
    ServiceId INT NOT NULL,
    PRIMARY KEY (DoctorId, ServiceId),
    FOREIGN KEY (DoctorId) REFERENCES Doctors(Id),
    FOREIGN KEY (ServiceId) REFERENCES VetServices(Id)
);
CREATE TABLE Diagnoses (
    Id VARCHAR(5) NOT NULL PRIMARY KEY,
    NameDiagnosis VARCHAR(255) NOT NULL UNIQUE
);
CREATE TABLE MedicalRecords (
    Id INT NOT NULL PRIMARY KEY,
    AppointmentsId INT NOT NULL,
    Diagnosis VARCHAR(5) NOT NULL,
    Weight DECIMAL(5,2) NOT NULL,
    Temperature DECIMAL(4,1) NOT NULL,
    Symptoms VARCHAR(255) NULL,
    Purpose VARCHAR(255) NOT NULL,

```

```
FOREIGN KEY (AppointmentsId) REFERENCES Appointments(Id),  
FOREIGN KEY (Diagnosis) REFERENCES Diagnoses(Id)  
);
```

Приложение В

(обязательное)

Исходный код серверной части приложения

Листинг В.1 – Файл AdminsController.cs

```

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AdminsController : ControllerBase
    {
        private readonly IAdminService _service;
        public AdminsController(IAdminService service)
        {
            _service = service;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<AdminDto>>>
        GetAll()
            => Ok(await _service.GetAllAsync());

        [HttpGet("{id}")]
        public async Task<ActionResult<AdminDto>> GetById(int id)
        {
            var admin = await _service.GetByIdAsync(id);
            if (admin == null) return NotFound();
            return Ok(admin);
        }

        [HttpPost]
        public async Task<IActionResult> Create([FromBody]
        AdminDto dto)
        {
            await _service.CreateAsync(dto);
            return CreatedAtAction(nameof(GetById), new { id =
            dto.Id }, dto);
        }

        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(int id)
        {
            await _service.DeleteAsync(id);
            return NoContent();
        }
    }
}

```

Листинг В.2 – Файл AppointmentsController.cs

```

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AppointmentsController : ControllerBase
    {
        private readonly IAppointmentService _service;
    }
}

```

```

    public AppointmentsController(IAppointmentService service)
    {
        _service = service;
    }

    [HttpGet("client/{clientId}")]
    public async
    Task<ActionResult<IEnumerable<AppointmentDto>>> GetByClientId(int
    clientId)
    {
        var appointments = await
        _service.GetByClientIdAsync(clientId);
        return Ok(appointments);
    }

    [HttpGet("doctor/{doctorId}")]
    public async
    Task<ActionResult<IEnumerable<AppointmentDto>>> GetByDoctorId(int
    doctorId)
    {
        var appointments = await
        _service.GetByDoctorIdAsync(doctorId);
        return Ok(appointments);
    }

    [HttpGet]
    public async
    Task<ActionResult<IEnumerable<AppointmentDto>>> GetAll()
        => Ok(await _service.GetAllAsync());

    [HttpGet("{id}")]
    public async Task<ActionResult<AppointmentDto>>
    GetById(int id)
    {
        var appointment = await _service.GetByIdAsync(id);
        if (appointment == null) return NotFound();
        return Ok(appointment);
    }

    [HttpPost]
    public async Task<IActionResult> Create([FromBody]
    AppointmentDto dto)
    {
        await _service.CreateAsync(dto);
        return CreatedAtAction(nameof(GetById), new { id =
        dto.Id }, dto);
    }

    [HttpPut("{id}")]
    public async Task<IActionResult> Update(int id, [FromBody]
    AppointmentDto dto)

```

```

        {
            await _service.UpdateAsync(id, dto);
            return NoContent();
        }

        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(int id)
        {
            await _service.DeleteAsync(id);
            return NoContent();
        }

        [HttpGet("available-dates")]
        public async Task<IActionResult>
        GetAvailableDates([FromQuery] int doctorId, [FromQuery] int
        serviceId)
        {
            var dates = await
            _service.GetAvailableDatesAsync(doctorId, serviceId);
            return Ok(dates);
        }

        [HttpGet("available-times")]
        public async Task<IActionResult>
        GetAvailableTimes([FromQuery] int doctorId, [FromQuery] int
        serviceId, [FromQuery] DateTime date)
        {
            var times = await
            _service.GetAvailableTimesAsync(doctorId, serviceId, date);
            return Ok(times);
        }

        [HttpPost("book")]
        public
        async Task<IActionResult> BookAppointment([FromBody]
        BookAppointmentDto dto)
        {
            var result = await _service.BookAppointmentAsync(dto);
            if (!result)
                return BadRequest("Запись не удалась. Время может
        быть занято.");

            return Ok("Запись успешно создана.");
        }
    }
}

```

Листинг В.3 – Файл AuthController.cs

```

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class AuthController : ControllerBase
    {

```

```

        private readonly IAuthService _authService;

        public AuthController(IAuthService authService)
        {
            _authService = authService;
        }
        [HttpPost("login")]
        public async Task<IActionResult> Login([FromBody]
LoginRequestDto dto)
        {
            try
            {
                var response = await _authService.LoginAsync(dto);
                return Ok(response);
            }
            catch (Exception ex)
            {
                return Unauthorized(ex.Message);
            }
        }
        [HttpPost("register")]
        public async Task<IActionResult> Register([FromBody]
RegisterRequestDto dto)
        {
            try
            {
                await _authService.RegisterAsync(dto);
                return Ok("Регистрация прошла успешно");
            }
            catch (Exception ex)
            {
                return BadRequest(ex.InnerException?.Message ??
ex.Message);
            }
        }
    }
}

```

Листинг В.4 – Файл ClientController.cs

```

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class ClientController : ControllerBase
    {
        private readonly IClientService _clientService;

        public ClientController(IClientService clientService)
        {
            _clientService = clientService;
        }

        [HttpGet]

```

```

        public async Task<ActionResult<IEnumerable<ClientDto>>>
GetAll()
    {
        var clients = await _clientService.GetAllAsync();
        return Ok(clients);
    }

    [HttpGet("{id}")]
    public async Task<ActionResult<ClientDto>> GetById(int id)
    {
        var client = await _clientService.GetByIdAsync(id);
        if (client == null) return NotFound();
        return Ok(client);
    }

    [HttpGet("user/{userId}")]
    public async Task<ActionResult<ClientDto>> GetByUserId(int
userId)
    {
        var client = await
_clientService.GetByUserIdAsync(userId);
        if (client == null) return NotFound();
        return Ok(client);
    }

    [HttpPost]
    public async Task<IActionResult> Add(ClientDto dto)
    {
        await _clientService.AddAsync(dto);
        return Ok();
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> Delete(int id)
    {
        await _clientService.DeleteAsync(id);
        return Ok();
    }
}
}

```

Листинг В.5 – Файл DoctorSchedulesController.cs

```

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class DoctorSchedulesController : ControllerBase
    {
        private readonly IDoctorScheduleService _service;

        public DoctorSchedulesController(IDoctorScheduleService
service)
        {

```

```

        _service = service;
    }

    [HttpGet]
    public async
Task<ActionResult<IEnumerable<DoctorScheduleDto>>> GetAll()
        => Ok(await _service.GetAllAsync());

    [HttpGet("{id}")]
    public async Task<ActionResult<DoctorScheduleDto>>
GetById(int id)
    {
        var schedule = await _service.GetByIdAsync(id);
        if (schedule == null) return NotFound();
        return Ok(schedule);
    }

    [HttpPost]
    public async Task<IActionResult> Create([FromBody]
DoctorScheduleDto dto)
    {
        await _service.CreateAsync(dto);
        return CreatedAtAction(nameof(GetById), new { id =
dto.Id }, dto);
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> Delete(int id)
    {
        await _service.DeleteAsync(id);
        return NoContent();
    }

    [HttpGet("doctor/{doctorId}/date/{date}")]
    public async Task<IActionResult>
GetSchedulesByDoctorAndDate(int doctorId, DateTime date)
    {
        var schedules = await
_service.GetSchedulesByDoctorAndDateAsync(doctorId, date);
        return Ok(schedules);
    }
}

```

Листинг В.6 – Файл DoctorsController.cs

```

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class DoctorsController : ControllerBase
    {
        private readonly IDoctorService _service;
        private readonly IDoctorVetServiceService
_doctorVetServiceService;
    }
}

```

```

        public DoctorsController(IDoctorService service,
IDoctorVetServiceService doctorVetServiceService)
        {
            _service = service;
            _doctorVetServiceService = doctorVetServiceService;
        }

        [HttpGet]
        public async Task<ActionResult<IEnumerable<DoctorDto>>>
GetAll()
            => Ok(await _service.GetAllAsync());

        [HttpGet("{id}")]
        public async Task<ActionResult<DoctorDto>> GetById(int id)
        {
            var doctor = await _service.GetByIdAsync(id);
            if (doctor == null) return NotFound();
            return Ok(doctor);
        }

        [HttpPost]
        public async Task<IActionResult> Create([FromBody]
DoctorDto dto)
        {
            await _service.CreateAsync(dto);
            return CreatedAtAction(nameof(GetById), new { id =
dto.Id }, dto);
        }
        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(int id)
        {
            await _service.DeleteAsync(id);
            return NoContent();
        }
        [HttpGet("by-service/{serviceId}")]
        public async Task<ActionResult<List<DoctorDto>>>
GetByServiceId(int serviceId)
        {
            var doctors = await
_doctorVetServiceService.GetByServiceIdAsync(serviceId);
            return Ok(doctors);
        }
    }
}

```

Листинг В.7 – Файл MedicalRecordsController.cs

```

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class MedicalRecordsController : ControllerBase
    {

```

```

        private readonly IMedicalRecordService _service;

        public MedicalRecordsController(IMedicalRecordService
service)
        {
            _service = service;
        }

        [HttpGet("{id}")]
        public async Task<ActionResult<MedicalRecordDto>> Get(int
id)
        {
            var record = await _service.GetByIdAsync(id);
            if (record == null)
                return NotFound();

            return Ok(record);
        }

        [HttpPost]
        public async Task<ActionResult<MedicalRecordDto>>
Post([FromBody] MedicalRecordCreatedDto dto)
        {
            var created = await _service.AddAsync(dto);
            return CreatedAtAction(nameof(Get), new { id =
created.Id }, created);
        }
    }
}

```

Листинг В.8 – Файл PetsController.cs

```

using Microsoft.AspNetCore.Mvc;
using VetClinicApi.DTOs;
using VetClinicApi.Models;
using VetClinicApi.Services;

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class PetsController : ControllerBase
    {
        private readonly IPetService _service;

        public PetsController(IPetService service)
        {
            _service = service;
        }

        [HttpGet("client/{clientId}")]
        public async Task<IActionResult> GetPetsByClientId(int
clientId)
        {

```

```

        var pets = await
_service.GetPetsByClientIdAsync(clientId);
        return Ok(pets);
    }

    [HttpGet("{id}")]
    public async Task<ActionResult<PetDto>> GetById(int id)
    {
        var pet = await _service.GetByIdAsync(id);
        if (pet == null) return NotFound();
        return Ok(pet);
    }

    [HttpPost]
    public async Task<IActionResult> Create([FromBody] PetDto
dto)
    {
        await _service.CreateAsync(dto);
        return CreatedAtAction(nameof(GetById), new { id =
dto.Id }, dto);
    }

    [HttpPut("{id}")]
    public async Task<IActionResult> Update(int id, [FromBody]
PetDto dto)
    {
        await _service.UpdateAsync(id, dto);
        return NoContent();
    }
}
}

```

Листинг В.9 – Файл RecommendationsController.cs

```

using Microsoft.AspNetCore.Mvc;
using VetClinicApi.DTOS;
using VetClinicApi.Services;

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class RecommendationsController : ControllerBase
    {
        private readonly IRecommendationService _service;

        public RecommendationsController(IRecommendationService
service)
        {
            _service = service;
        }

        [HttpGet("{id}")]

```

```

        public async Task<ActionResult<RecommendationDto>>
GetById(int id)
    {
        var dto = await _service.GetByIdAsync(id);
        if (dto == null) return NotFound();
        return Ok(dto);
    }

    [HttpPost]
    public async Task<IActionResult> Create([FromBody]
RecommendationDto dto)
    {
        await _service.CreateAsync(dto);
        return CreatedAtAction(nameof(GetById), new { id =
dto.Id }, dto);
    }
    [HttpDelete("{id}")]
    public async Task<IActionResult> Delete(int id)
    {
        await _service.DeleteAsync(id);
        return NoContent();
    }
}
}

```

Листинг В.10 – Файл UserController.cs

```

using Microsoft.AspNetCore.Mvc;
using VetClinicApi.DTOs;
using VetClinicApi.Services;

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class UserController : ControllerBase
    {
        private readonly IUserService _userService;

        public UserController(IUserService userService)
        {
            _userService = userService;
        }
        [HttpGet("{id}")]
        public async Task<ActionResult<UserDto>> GetById(int id)
        {
            var user = await _userService.GetByIdAsync(id);
            if (user == null)
                return NotFound();

            return Ok(user);
        }

        [HttpGet("login/{login}")]

```

```

        public async Task<ActionResult<UserDto>> GetByLogin(string
login)
        {
            var user = await _userService.GetByLoginAsync(login);
            if (user == null)
                return NotFound();

            return Ok(user);
        }

        [HttpPost]
        public async Task<ActionResult> Create(UserDto userDto)
        {
            await _userService.CreateAsync(userDto);
            return CreatedAtAction(nameof(GetByEmail), new { email
= userDto.Email }, userDto);
        }
        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(int id)
        {
            await _userService.DeleteAsync(id);
            return NoContent();
        }
    }
}

```

Листинг В.11 – Файл VetServicesController.cs

```

using VetClinicApi.Services;

namespace VetClinicApi.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class VetServicesController : ControllerBase
    {
        private readonly IVetServiceService _service;

        public VetServicesController(IVetServiceService service)
        {
            _service = service;
        }

        [HttpGet]
        public async
Task<ActionResult<IEnumerable<VetServiceDto>>> GetAll()
            => Ok(await _service.GetAllAsync());

        [HttpGet("{id}")]
        public async Task<ActionResult<VetServiceDto>> GetById(int
id)
        {
            var service = await _service.GetByIdAsync(id);
            if (service == null) return NotFound();
        }
    }
}

```

```

        return Ok(service);
    }

    [HttpPost]
    public async Task<IActionResult> Create([FromBody]
VetServiceDto dto)
    {
        await _service.CreateAsync(dto);
        return CreatedAtAction(nameof(GetById), new { id =
dto.Id }, dto);
    }

    [HttpPut("{id}")]
    public async Task<IActionResult> Update(int id, [FromBody]
VetServiceDto dto)
    {
        await _service.UpdateAsync(id, dto);
        return NoContent();
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> Delete(int id)
    {
        await _service.DeleteAsync(id);
        return NoContent();
    }
}
}

```

Листинг В.12 – Файл AppointmentDto.cs

```

using VetClinicApi.Models;

namespace VetClinicApi.DTOs
{
    public class AppointmentDto
    {
        public int Id { get; init; }
        public int ClientId { get; set; }
        public int PetId { get; set; }
        public string? PetsName { get; set; }
        public int ServiceId { get; set; }
        public string? ServiceName { get; set; }
        public int DoctorId { get; set; }
        public string? LastName { get; set; } = null!;
        public string? FirstName { get; set; } = null!;
        public int SchedulId { get; set; }
        public DateTime? AvailableDate { get; set; }
        public TimeSpan? AvailableTime { get; set; }
        public string? AppointmentStatus { get; set; }
        public int? RecommendationId { get; set; }
        public string? Recommendation { get; set; }
    }
}

```

Листинг В.13 – Файл BookAppointmentDto.cs

```
namespace VetClinicApi.DTOs
{
    public class BookAppointmentDto
    {
        public int DoctorId { get; set; }
        public int ServiceId { get; set; }
        public int PetId { get; set; }
        public DateTime AvailableDate { get; set; }
        public TimeSpan AvailableTime { get; set; }

        public int ClientId { get; set; }
    }
}
```

Листинг В.14 – Файл ClientDto.cs

```
namespace VetClinicApi.DTOs
{
    public class ClientDto
    {
        public int Id { get; init; }
        public int UserId { get; set; }
        public string LastName { get; set; }
        public string FirstName { get; set; }
        public string? MiddleName { get; set; }
        public string PhoneNumber { get; set; }
        public string? Email { get; set; }
    }
}
```

Листинг В.15 – Файл DoctorDto.cs

```
namespace VetClinicApi.DTOs
{
    public class DoctorDto
    {
        public int Id { get; init; }
        public int? UserId { get; set; }
        public string LastName { get; set; }
        public string FirstName { get; set; }
        public string? MiddleName { get; set; }
        public string? Specialization { get; set; }
        public string? PhoneNumber { get; set; }
        public string? Email { get; set; }
    }
}
```

Листинг В.16 – Файл DoctorScheduleDto.cs

```
namespace VetClinicApi.DTOs
{
    public class DoctorScheduleDto
    {
```

```

        public int Id { get; init; }
        public int DoctorId { get; set; }
        public DateTime AvailableDate { get; set; }
        public TimeSpan AvailableTime { get; set; }
        public bool IsBooked { get; set; }
    }
}

```

Листинг В.17 – Файл MedicalRecordDto.cs

```

namespace VetClinicApi.DTOs
{
    public class MedicalRecordDto
    {
        public int Id { get; set; }
        public int ApointmentsId { get; set; }
        public string? Diagnosis { get; set; }
        public double? Weight { get; set; }
        public double? Temperature { get; set; }
        public string? Symptoms { get; set; }
        public string Purpose { get; set; } = null!;
    }
}

```

Листинг В.18 – Файл LoginRequestDto.cs

```

namespace VetClinicApi.DTOs
{
    public class LoginRequestDto
    {
        public string Login { get; set; } = null!;
        public string Password { get; set; } = null!;
    }
}

```

Листинг В.19 – Файл LoginResponseDto.cs

```

namespace VetClinicApi.DTOs
{
    public class LoginResponseDto
    {
        public string Role { get; set; } = null!;
        public int UserId { get; set; }
        public int? ClientId { get; set; }
        public int? DoctorId { get; set; }
        public int? AdminId { get; set; }
    }
}

```

Листинг В.20 – Файл PetDto.cs

```

namespace VetClinicApi.DTOs
{
    public class PetDto
    {

```

```

        public int Id { get; init; }
        public int ClientId { get; set; }
        public string PetsName { get; set; }
        public string? PetsType { get; set; }
        public string? Breed { get; set; }
        public DateTime? BirthDate { get; set; }
        public string? Gender { get; set; }
    }
}

```

Листинг В.21 – Файл RecommendationDto.cs

```

namespace VetClinicApi.DTOs
{
    public class RecommendationDto
    {
        public int Id { get; init; }
        public int DoctorId { get; set; }
        public string? RecommendText { get; set; }
        public DateTime? RecommendDate { get; set; }
    }
}

```

Листинг В.22 – Файл RegisterRequestDto.cs

```

namespace VetClinicApi.DTOs
{
    public class RegisterRequestDto
    {
        public string Email { get; set; } = null!;
        public string Password { get; set; } = null!;
        public string LastName { get; set; } = null!;
        public string FirstName { get; set; } = null!;
        public string MiddleName { get; set; } = null!;
        public string PhoneNumber { get; set; } = null!;
    }
}

```

Листинг В.23 – Файл VetServiceDto.cs

```

namespace VetClinicApi.DTOs
{
    public class VetServiceDto
    {
        public int Id { get; init; }
        public string ServicName { get; set; }
        public string? ServicDescription { get; set; }
        public decimal Price { get; set; }
    }
}

```

Листинг В.24 – Файл Admin.cs

```

using Microsoft.EntityFrameworkCore;

```

```

namespace VetClinicApi.Models
{
    public class Admin
    {
        public int Id { get; set; }
        public int? UserId { get; set; }

        public string LastName { get; set; } = null!;
        public string FirstName { get; set; } = null!;
        public string? MiddleName { get; set; }
        public string? PhoneNumber { get; set; }
        public string? Email { get; set; }

        public User? User { get; set; }
    }
}

```

Листинг В.25 – Файл Appointment.cs

```

using System.ComponentModel.DataAnnotations.Schema;

namespace VetClinicApi.Models
{
    public class Appointment
    {
        public int Id { get; set; }
        public int ClientId { get; set; }
        public int PetId { get; set; }
        public int ServiceId { get; set; }
        public int DoctorId { get; set; }

        [ForeignKey("Schedule")]
        public int SchedulId { get; set; }
        public string? AppointmentStatus { get; set; }
        public int? RecommendationId { get; set; }

        public Client? Client { get; set; }
        public Pet? Pet { get; set; }
        public VetService? Service { get; set; }
        public Doctor? Doctor { get; set; }
        public DoctorSchedule? Schedule { get; set; }
        public Recommendation? Recommendation { get; set; }
    }
}

```

Листинг В.26 – Файл Client.cs

```

using Microsoft.EntityFrameworkCore;

namespace VetClinicApi.Models
{
    [Index(nameof(Email), IsUnique = true)]
    [Index(nameof(PhoneNumber), IsUnique = true)]
    [Index(nameof(UserId), IsUnique = true)]

```

```

public class Client
{
    public int Id { get; set; }
    public int UserId { get; set; }

    public string LastName { get; set; } = null!;
    public string FirstName { get; set; } = null!;
    public string? MiddleName { get; set; }
    public string PhoneNumber { get; set; } = null!;
    public string? Email { get; set; }

    public User? User { get; set; }
}

```

Листинг В.27 – Файл Doctor.cs

```

using Microsoft.EntityFrameworkCore;

namespace VetClinicApi.Models
{
    [Index(nameof(Email), IsUnique = true)]
    [Index(nameof(PhoneNumber), IsUnique = true)]
    [Index(nameof(UserId), IsUnique = true)]
    public class Doctor
    {
        public int Id { get; set; }
        public int? UserId { get; set; }

        public string LastName { get; set; } = null!;
        public string FirstName { get; set; } = null!;
        public string? MiddleName { get; set; }
        public string? Specialization { get; set; }
        public string? PhoneNumber { get; set; }
        public string? Email { get; set; }

        public User? User { get; set; }

        public ICollection<DoctorVetService> DoctorVetServices {
get; set; }
    }
}

```

Листинг В.28 – Файл DoctorSchedule.cs

```

namespace VetClinicApi.Models
{
    public class DoctorSchedule
    {
        public int Id { get; set; }
        public int DoctorId { get; set; }

        public DateTime AvailableDate { get; set; }
        public TimeSpan AvailableTime { get; set; }
    }
}

```

```

        public bool IsBooked { get; set; } = false;

        public Doctor? Doctor { get; set; }
    }
}

```

Листинг В.29 – Файл DoctorVetService.cs

```

namespace VetClinicApi.Models
{
    public class DoctorVetService
    {
        public int DoctorId { get; set; }
        public Doctor Doctor { get; set; }

        public int ServiceId { get; set; }
        public VetService VetService { get; set; }
    }
}

```

Листинг В.30 – Файл MedicalRecords.cs

```

namespace VetClinicApi.Models
{
    public class MedicalRecords
    {
        public int Id { get; set; }
        public int ApointmentsId { get; set; }
        public string? Diagnosis { get; set; } = null!;
        public double? Weight { get; set; }
        public double? Temperature { get; set; }
        public string? Symptoms { get; set; }
        public string Purpose { get; set; }

    }
}

```

Листинг В.31 – Файл Pet.cs

```

namespace VetClinicApi.Models
{
    public class Pet
    {
        public int Id { get; set; }
        public int ClientId { get; set; }
        public string PetsName { get; set; } = null!;
        public string? PetsType { get; set; }
        public string? Breed { get; set; }
        public DateTime? BirthDate { get; set; }
        public string? Gender { get; set; }

        public Client? Client { get; set; }
    }
}

```

Листинг В.32 – Файл Recommendation.cs

```
namespace VetClinicApi.Models
{
    public class Recommendation
    {
        public int Id { get; set; }
        public int DoctorId { get; set; }

        public string? RecommendText { get; set; }
        public DateTime? RecommendDate { get; set; }

        public Doctor? Doctor { get; set; }
    }
}
```

Листинг В.33 – Файл User.cs

```
using System.Security.Claims;

namespace VetClinicApi.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Login { get; set; } = null!;
        public string Password { get; set; } = null!;
        public string UserRole { get; set; } = null!; // Client,
        Doctor, Admin

    }
}
```

Листинг В.34 – Файл VetService.cs

```
namespace VetClinicApi.Models
{
    public class VetService
    {
        public int Id { get; set; }
        public string ServicName { get; set; } = null!;
        public string? ServicDescription { get; set; }
        public decimal Price { get; set; }
        public ICollection<DoctorVetService> DoctorVetServices {
get; set; }
    }
}
```

Листинг В.35 – Файл AdminRepository.cs

```
using Microsoft.EntityFrameworkCore;
using VetClinicApi.Data;
using VetClinicApi.Models;
```

```

namespace VetClinicApi.Repositories
{
    public class AdminRepository : IAdminRepository
    {
        private readonly ApplicationDbContext _context;

        public AdminRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<Admin?> GetByIdAsync(int id) =>
            await _context.Admins.FindAsync(id);

        public async Task DeleteAsync(int id)
        {
            var admin = await GetByIdAsync(id);
            if (admin != null)
            {
                _context.Admins.Remove(admin);
                await _context.SaveChangesAsync();
            }
        }
    }
}

```

Листинг В.36 – Файл AppointmentRepository.cs

```

using VetClinicApi.Data;
using VetClinicApi.Models;
using Microsoft.EntityFrameworkCore;
using VetClinicApi.DTOs;

namespace VetClinicApi.Repositories
{
    public class AppointmentRepository : IAppointmentRepository
    {
        private readonly ApplicationDbContext _context;

        public AppointmentRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<IEnumerable<Appointment>>
        GetByIdClientIdAsync(int clientId)
        {
            return await _context.Appointments
                .Where(a => a.ClientId == clientId)
                .Include(a => a.Pet)
                .Include(a => a.Service)
                .Include(a => a.Doctor)
                .Include(a => a.Schedule)
                .Include(a => a.Recommendation)
                .ToListAsync();
        }
    }
}

```

```

    }

    public async Task<IEnumerable<Appointment>>
    GetByDoctorIdAsync(int doctorId)
    {
        return await _context.Appointments
            .Where(a => a.DoctorId == doctorId)
            .Include(a => a.Pet)
            .Include(a => a.Service)
            .Include(a => a.Client)
            .Include(a => a.Schedule)
            .Include(a => a.Recommendation)
            .ToListAsync();
    }

    public async Task<Appointment?> GetByIdAsync(int id) =>
        await _context.Appointments.FindAsync(id);

    public async Task DeleteAsync(int id)
    {
        var appointment = await _context.Appointments
            .Include(a => a.Schedule) // Важно для получения
расписания
            .FirstOrDefaultAsync(a => a.Id == id);

        if (appointment != null)
        {
            // Освобождаем слот расписания
            if (appointment.Schedule != null)
            {
                appointment.Schedule.IsBooked = false;
            }

            _context.Appointments.Remove(appointment);
            await _context.SaveChangesAsync();
        }
    }

    public async Task<IEnumerable<DateTime>>
    GetAvailableDatesAsync(int doctorId, int serviceId)
    {
        var today = DateTime.Today;

        var vetService = await _context.DoctorVetServices
            .FirstOrDefaultAsync(v => v.DoctorId == doctorId
&& v.ServiceId == serviceId);

        if (vetService == null)
            return Enumerable.Empty<DateTime>();

        return await _context.DoctorSchedules
            .Where(s => s.DoctorId == doctorId
                && s.AvailableDate.Date >= today
                && !s.IsBooked)
            .Select(s => s.AvailableDate.Date)

```

```

        .Distinct()
        .OrderBy(d => d)
        .ToListAsync();
    }

    public async Task<IEnumerable<TimeSpan>>
    GetAvailableTimesAsync(int doctorId, int serviceId, DateTime date)
    {
        var vetService = await _context.DoctorVetServices
            .FirstOrDefaultAsync(v => v.DoctorId == doctorId &&
v.ServiceId == serviceId);

        if (vetService == null)
            return Enumerable.Empty<TimeSpan>();

        return await _context.DoctorSchedules
            .Where(s => s.DoctorId == doctorId
                && s.AvailableDate.Date == date.Date
                && !s.IsBooked)
            .Select(s => s.AvailableTime)
            .OrderBy(t => t)
            .ToListAsync();
    }

    public async Task<bool>
    BookAppointmentAsync(BookAppointmentDto dto)
    {
        // Ищем расписание, подходящее под врача, дату и время
        var schedule = await
        _context.DoctorSchedules.FirstOrDefaultAsync(s =>
            s.DoctorId == dto.DoctorId &&
            s.AvailableDate.Date == dto.AvailableDate.Date &&
            s.AvailableTime == dto.AvailableTime &&
            !s.IsBooked);

        if (schedule == null)
            return false; // Нет доступного слота

        var appointment = new Appointment
        {
            DoctorId = dto.DoctorId,
            ClientId = dto.ClientId,
            PetId = dto.PetId,
            ServiceId = dto.ServiceId,
            SchedulId = schedule.Id, // важно!
            AppointmentStatus = "Запланировано"
        };

        _context.Appointments.Add(appointment);

        // Отметим слот как занятый
        schedule.IsBooked = true;

        await _context.SaveChangesAsync();
    }

```

```

        return true;
    }
}

```

Листинг В.37 – Файл DoctorRepository.cs

```

namespace VetClinicApi.Repositories
{
    public class DoctorRepository : IDoctorRepository
    {
        private readonly ApplicationDbContext _context;

        public DoctorRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<Doctor?> GetByIdAsync(int id) =>
            await _context.Doctors.FindAsync(id);

        public async Task AddAsync(Doctor doctor)
        {
            await _context.Doctors.AddAsync(doctor);
            await _context.SaveChangesAsync();
        }

        public async Task DeleteAsync(int id)
        {
            var doctor = await GetByIdAsync(id);
            if (doctor != null)
            {
                _context.Doctors.Remove(doctor);
                await _context.SaveChangesAsync();
            }
        }
    }
}

```

Листинг В.38 – Файл DoctorScheduleRepository.cs

```

using VetClinicApi.Data;
using VetClinicApi.Models;
using Microsoft.EntityFrameworkCore;

namespace VetClinicApi.Repositories
{
    public class DoctorScheduleRepository :
        IDoctorScheduleRepository
    {
        private readonly ApplicationDbContext _context;

        public DoctorScheduleRepository(ApplicationDbContext
context)
        {
            _context = context;
        }
    }
}

```

```

    }
    public async Task<DoctorSchedule?> GetByIdAsync(int id) =>
        await _context.DoctorSchedules.FindAsync(id);

    public async Task AddAsync(DoctorSchedule schedule)
    {
        await _context.DoctorSchedules.AddAsync(schedule);
        await _context.SaveChangesAsync();
    }

    public async Task DeleteAsync(int id)
    {
        var schedule = await GetByIdAsync(id);
        if (schedule != null)
        {
            _context.DoctorSchedules.Remove(schedule);
            await _context.SaveChangesAsync();
        }
    }

    public async Task<IEnumerable<DoctorSchedule>>
    GetSchedulesByDoctorAndDateAsync(int doctorId, DateTime date)
    {
        return await _context.DoctorSchedules
            .Where(ds => ds.DoctorId == doctorId &&
            ds.AvailableDate.Date == date.Date)
            .ToListAsync();
    }
}

```

Листинг В.39 – Файл PetRepository.cs

```

using VetClinicApi.Models;

namespace VetClinicApi.Repositories
{
    public class PetRepository : IPetRepository
    {
        private readonly ApplicationDbContext _context;

        public PetRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<IEnumerable<Pet>> GetAllAsync() =>
            await _context.Pets.ToListAsync();

        public async Task<Pet?> GetByIdAsync(int id) =>
            await _context.Pets.FindAsync(id);

        public async Task AddAsync(Pet pet)
        {
            await _context.Pets.AddAsync(pet);
        }
    }
}

```

```

        await _context.SaveChangesAsync();
    }

    public async Task UpdateAsync(Pet pet)
    {
        _context.Pets.Update(pet);
        await _context.SaveChangesAsync();
    }

    public async Task<IEnumerable<Pet>>
GetPetsByClientIdAsync(int clientId)
    {
        return await _context.Pets
            .Where(p => p.ClientId == clientId)
            .ToListAsync();
    }
}

```

Листинг В.40 – Файл UserRepository.cs

```

using VetClinicApi.Data;
using VetClinicApi.Models;

namespace VetClinicApi.Repositories
{
    public class UserRepository : IUserRepository
    {
        private readonly ApplicationDbContext _context;

        public UserRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<User?> GetByIdAsync(int id)
        {
            return await _context.Users.FindAsync(id);
        }

        public async Task<User?> GetByEmailAsync(string email)
        {
            return await _context.Users.FirstOrDefault(u =>
u.Email == email);
        }

        public async Task AddAsync(User user)
        {
            await _context.Users.AddAsync(user);
            await _context.SaveChangesAsync();
        }

        public async Task DeleteAsync(int id)
        {
            var user = await _context.Users.FindAsync(id);
            if (user != null)
            {

```

```

        _context.Users.Remove(user);
        await _context.SaveChangesAsync();
    }
}
}

```

Листинг В.41 – Файл VetServiceRepository.cs

```

using VetClinicApi.Data;
namespace VetClinicApi.Repositories
{
    public class VetServiceRepository : IVetServiceRepository
    {
        private readonly ApplicationDbContext _context;

        public VetServiceRepository(ApplicationDbContext context)
        {
            _context = context;
        }

        public async Task<IEnumerable<VetService>> GetAllAsync()
=>
            await _context.VetServices.ToListAsync();

        public async Task<VetService?> GetByIdAsync(int id) =>
            await _context.VetServices.FindAsync(id);

        public async Task AddAsync(VetService service)
        {
            await _context.VetServices.AddAsync(service);
            await _context.SaveChangesAsync();
        }

        public async Task UpdateAsync(VetService service)
        {
            _context.VetServices.Update(service);
            await _context.SaveChangesAsync();
        }

        public async Task DeleteAsync(int id)
        {
            var service = await GetByIdAsync(id);
            if (service != null)
            {
                _context.VetServices.Remove(service);
                await _context.SaveChangesAsync();
            }
        }
    }
}

```

Листинг В.43 – Файл AppointmentService.cs

```

using VetClinicApi.Repositories;

```

```

using static System.Runtime.InteropServices.JavaScript.JSType;

namespace VetClinicApi.Services
{
    public class AppointmentService : IAppointmentService
    {
        private readonly IAppointmentRepository _repository;

        public AppointmentService(IAppointmentRepository repository)
        {
            _repository = repository;
        }

        public async Task<IEnumerable<AppointmentDto>>
        GetByClientIdAsync(int clientId)
        {
            var appointments = await
            _repository.GetByClientIdAsync(clientId);
            return appointments.Select(AppointmentMapper.ToDto);
        }

        public async Task<IEnumerable<AppointmentDto>>
        GetByDoctorIdAsync(int doctorId)
        {
            var appointments = await
            _repository.GetByDoctorIdAsync(doctorId);
            return appointments.Select(AppointmentMapper.ToDto);
        }

        public async Task<AppointmentDto?> GetByIdAsync(int id)
        {
            var appointment = await _repository.GetByIdAsync(id);
            return AppointmentMapper.ToDto(appointment);
        }

        public async Task DeleteAsync(int id)
        {
            await _repository.DeleteAsync(id);
        }

        public async Task<IEnumerable<DateTime>>
        GetAvailableDatesAsync(int doctorId, int serviceId)
        {
            return await
            _repository.GetAvailableDatesAsync(doctorId, serviceId);
        }

        public async Task<IEnumerable<TimeSpan>>
        GetAvailableTimesAsync(int doctorId, int serviceId, DateTime date)
        {
            return await
            _repository.GetAvailableTimesAsync(doctorId, serviceId, date);
        }

        public async Task<bool>
        BookAppointmentAsync(BookAppointmentDto dto)
    }
}

```

```

        {
            return await _repository.BookAppointmentAsync(dto);
        }
    }
}

```

Листинг В.44 – Файл AuthService.cs

```

using VetClinicApi.DTOs;
using VetClinicApi.Models;
using System.Text;
using VetClinicApi.Data;

namespace VetClinicApi.Services
{
    public class AuthService : IAuthService
    {
        private readonly ApplicationDbContext _context;
        private readonly IConfiguration _configuration;

        public AuthService(ApplicationDbContext context,
            IConfiguration configuration)
        {
            _context = context;
            _configuration = configuration;
        }

        public async Task RegisterAsync(RegisterRequestDto dto)
        {
            var userExists = await _context.Users.AnyAsync(u =>
u.Email == dto.Email);
            if (userExists)
                throw new Exception("Пользователь с таким Email
уже существует.");
            var user = new User
            {
                Email = dto.Email,
                PasswordHash = dto.Password,
                UserRole = "Client"
            };
            _context.Users.Add(user);
            await _context.SaveChangesAsync();
            string role;

            var doctor = await
_context.Doctors.FirstOrDefaultAsync(d => d.Email == dto.Email);
            if (doctor != null)
            {
                doctor.UserId = user.Id;
                role = "Doctor";
            }
            else
            {

```

```

        var admin = await
_context.Admins.FirstOrDefaultAsync(a => a.Email == dto.Email);
        if (admin != null)
        {
            admin.UserId = user.Id;
            role = "Admin";
        }
        else
        {
            role = "Client";
            var client = new Client
            {
                FirstName = dto.FirstName,
                LastName = dto.LastName,
                MiddleName = dto.MiddleName,
                PhoneNumber = dto.PhoneNumber,
                Email = dto.Email,
                UserId = user.Id
            };
            _context.Clients.Add(client);
        }
        user.UserRole = role;
        await _context.SaveChangesAsync();
    }
    public async Task<LoginResponseDto>
LoginAsync(LoginRequestDto dto)
    {
        var user = await _context.Users.FirstOrDefaultAsync(u
=> u.Email == dto.Email);
        if (user == null || user.PasswordHash != dto.Password)
            throw new Exception("Неверный email или пароль.");

        var claims = new[]
        {
            new Claim(ClaimTypes.Name, user.Email),
            new Claim(ClaimTypes.Role, user.UserRole)
        };

        var key = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Ke
y"]));
        var creds = new SigningCredentials(key,
SecurityAlgorithms.HmacSha256);

        var token = new JwtSecurityToken(
            issuer: _configuration["Jwt:Issuer"],
            audience: _configuration["Jwt:Audience"],
            claims: claims,
            expires: DateTime.Now.AddHours(3),
            signingCredentials: creds
        );
    }

```

```

        var response = new LoginResponseDto
        {
            Token = new
JwtSecurityTokenHandler().WriteToken(token),
            Role = user.UserRole,
            UserId = user.Id,
        };
        if (user.UserRole == "Client")
        {
            var client = await
_context.Clients.FirstOrDefaultAsync(c => c.UserId == user.Id);
            response.ClientId = client?.Id;
        }
        if (user.UserRole == "Doctor")
        {
            var doctor = await
_context.Doctors.FirstOrDefaultAsync(c => c.UserId == user.Id);
            response.DoctorId = doctor?.Id;
        }
        if (user.UserRole == "Admin")
        {
            var admin = await
_context.Admins.FirstOrDefaultAsync(c => c.UserId == user.Id);
            response.AdminId = admin?.Id;
        }
        return response;
    }
}
}

```

Листинг В.45 – Файл ClientService.cs

```

using VetClinicApi.DTOs;
using VetClinicApi.Mappers;
using VetClinicApi.Repositories;

namespace VetClinicApi.Services
{
    public class ClientService : IClientService
    {
        private readonly IClientRepository _clientRepository;

        public ClientService(IClientRepository clientRepository)
        {
            _clientRepository = clientRepository;
        }

        public async Task<IEnumerable<ClientDto>> GetAllAsync()
        {
            var clients = await _clientRepository.GetAllAsync();
            return clients.Select(ClientMapper.ToDto);
        }

        public async Task<ClientDto?> GetByIdAsync(int id)
        {

```

```

        var client = await _clientRepository.GetByIdAsync(id);
        return ClientMapper.ToDto(client);
    }

    public async Task<ClientDto?> GetByUserIdAsync(int userId)
    {
        var client = await
_clientRepository.GetByUserIdAsync(userId);
        return ClientMapper.ToDto(client);
    }

    public async Task AddAsync(ClientDto clientDto)
    {
        var client = ClientMapper.ToEntity(clientDto);
        await _clientRepository.AddAsync(client);
    }

    public async Task UpdateAsync(ClientDto clientDto)
    {
        var existing = await
_clientRepository.GetByUserIdAsync(clientDto.UserId);
        if (existing == null)
            throw new Exception("Client not found");

        existing.FirstName = clientDto.FirstName;
        existing.LastName = clientDto.LastName;
        existing.MiddleName = clientDto.MiddleName;
        existing.PhoneNumber = clientDto.PhoneNumber;
        existing.Email = clientDto.Email;

        await _clientRepository.UpdateAsync(existing);
    }
}

```

Листинг В.46 – Файл DoctorScheduleService.cs

```

using VetClinicApi.DTOs;
using VetClinicApi.Mappers;
using VetClinicApi.Models;
using VetClinicApi.Repositories;

namespace VetClinicApi.Services
{
    public class DoctorScheduleService : IDoctorScheduleService
    {
        private readonly IDoctorScheduleRepository _repository;

        public DoctorScheduleService(IDoctorScheduleRepository
repository)
        {
            _repository = repository;
        }
    }
}

```

```

public async Task<DoctorScheduleDto?> GetByIdAsync(int id)
{
    var schedule = await _repository.GetByIdAsync(id);
    return DoctorScheduleMapper.ToDto(schedule);
}

public async Task CreateAsync(DoctorScheduleDto dto)
{
    var entity = DoctorScheduleMapper.ToEntity(dto);
    await _repository.AddAsync(entity);
}

public async Task UpdateAsync(int id, DoctorScheduleDto
dto)
{
    var entity = DoctorScheduleMapper.ToEntity(dto);
    entity.Id = id;
    await _repository.UpdateAsync(entity);
}

public async Task DeleteAsync(int id)
{
    await _repository.DeleteAsync(id);
}

public async Task<IEnumerable<DoctorSchedule>>
GetSchedulesByDoctorAndDateAsync(int doctorId, DateTime date)
{
    return await
_repository.GetSchedulesByDoctorAndDateAsync(doctorId, date);
}
}

```

Листинг В.47 – Файл DoctorService.cs

```

using Microsoft.EntityFrameworkCore;
using VetClinicApi.Models;
using VetClinicApi.Repositories;

namespace VetClinicApi.Services
{
    public class DoctorService : IDoctorService
    {
        private readonly IDoctorRepository _repository;

        public DoctorService(IDoctorRepository repository)
        {
            _repository = repository;
        }

        public async Task<IEnumerable<DoctorDto>> GetAllAsync()
        {
            var doctors = await _repository.GetAllAsync();
            return doctors.Select(DoctorMapper.ToDto);
        }
    }
}

```

```

    }

    public async Task<DoctorDto?> GetByIdAsync(int id)
    {
        var doctor = await _repository.GetByIdAsync(id);
        return DoctorMapper.ToDto(doctor);
    }

    public async Task CreateAsync(DoctorDto dto)
    {
        var doctor = DoctorMapper.ToEntity(dto);
        await _repository.AddAsync(doctor);
    }

    public async Task DeleteAsync(int id) =>
        await _repository.DeleteAsync(id);
    }
}

```

Листинг В.48 – Файл DoctorVetServiceService.cs

```

using VetClinicApi.DTOs;
using VetClinicApi.Models;

namespace VetClinicApi.Services
{
    public interface IDoctorVetServiceService
    {
        Task<IEnumerable<DoctorVetServiceDto>> GetAllAsync();
        Task AddAsync(DoctorVetServiceDto dto);
        Task DeleteAsync(DoctorVetServiceDto dto);
        Task<IEnumerable<DoctorVetService>> GetByDoctorIdAsync(int
doctorId);

        Task<List<DoctorDto>> GetByServiceIdAsync(int serviceId);
    }

    public class DoctorVetServiceService :
IDoctorVetServiceService
    {
        private readonly ApplicationDbContext _context;

        public DoctorVetServiceService(ApplicationDbContext
context)
        {
            _context = context;
        }

        public async Task<IEnumerable<DoctorVetServiceDto>>
GetAllAsync()
        {
            return _context.DoctorVetServices
                .Select(dvs => new DoctorVetServiceDto
                {
                    DoctorId = dvs.DoctorId,

```

```

        ServiceId = dvs.ServiceId
    });
}
public async Task AddAsync(DoctorVetServiceDto dto)
{
    var exists = await
_context.DoctorVetServices.FindAsync(dto.DoctorId, dto.ServiceId);
    if (exists == null)
    {
        _context.DoctorVetServices.Add(new
DoctorVetService
        {
            DoctorId = dto.DoctorId,
            ServiceId = dto.ServiceId
        });
        await _context.SaveChangesAsync();
    }
}
public async Task DeleteAsync(DoctorVetServiceDto dto)
{
    var entity = await
_context.DoctorVetServices.FindAsync(dto.DoctorId, dto.ServiceId);
    if (entity != null)
    {
        _context.DoctorVetServices.Remove(entity);
        await _context.SaveChangesAsync();
    }
}
public async Task<IEnumerable<DoctorVetService>>
GetByDoctorIdAsync(int doctorId)
{
    return await _context.DoctorVetServices
        .Where(dvs => dvs.DoctorId == doctorId)
        .ToListAsync();
}
public async Task<List<DoctorDto>> GetByServiceIdAsync(int
serviceId)
{
    var doctors = await _context.DoctorVetServices
        .Where(dvs => dvs.ServiceId == serviceId)
        .Include(dvs => dvs.Doctor)
        .Select(dvs => dvs.Doctor)
        .Distinct()
        .ToListAsync();

    return doctors.Select(d => new DoctorDto
    {
        Id = d.Id,
        LastName = d.LastName,
        FirstName = d.FirstName,
        Specialization = d.Specialization
    }).ToList();
}

```

```
}
```

Листинг В.49 – Файл PetService.cs

```
using VetClinicApi.DTOs;
using VetClinicApi.Mappers;
using VetClinicApi.Models;
using VetClinicApi.Repositories;

namespace VetClinicApi.Services
{
    public class PetService : IPetService
    {
        private readonly IPetRepository _repository;

        public PetService(IPetRepository repository)
        {
            _repository = repository;
        }

        public async Task<IEnumerable<PetDto>> GetAllAsync()
        {
            var pets = await _repository.GetAllAsync();
            return pets.Select(PetMapper.ToDto);
        }

        public async Task<PetDto?> GetByIdAsync(int id)
        {
            var pet = await _repository.GetByIdAsync(id);
            return PetMapper.ToDto(pet);
        }

        public async Task CreateAsync(PetDto dto)
        {
            var pet = PetMapper.ToEntity(dto);
            await _repository.AddAsync(pet);
        }

        public async Task UpdateAsync(int id, PetDto dto)
        {
            var pet = PetMapper.ToEntity(dto);
            pet.Id = id;
            await _repository.UpdateAsync(pet);
        }

        public async Task DeleteAsync(int id)
        {
            await _repository.DeleteAsync(id);
        }

        public async Task<IEnumerable<Pet>>
GetPetsByClientIdAsync(int clientId)
        {
            return await
_repository.GetPetsByClientIdAsync(clientId);
        }
    }
}
```

Листинг В.50 – Файл RecommendationService.cs

```

using VetClinicApi.DTOs;
using VetClinicApi.Mappers;
using VetClinicApi.Repositories;

namespace VetClinicApi.Services
{
    public class RecommendationService : IRecommendationService
    {
        private readonly IRecommendationRepository _repository;

        public RecommendationService(IRecommendationRepository repository)
        {
            _repository = repository;
        }

        public async Task<RecommendationDto?> GetByIdAsync(int id)
        {
            var entity = await _repository.GetByIdAsync(id);
            return RecommendationMapper.ToDto(entity);
        }

        public async Task CreateAsync(RecommendationDto dto)
        {
            var entity = RecommendationMapper.ToEntity(dto);
            await _repository.AddAsync(entity);
        }
    }
}

```

Листинг В.51 – Файл UserService.cs

```

using VetClinicApi.DTOs;
using VetClinicApi.Mappers;
using VetClinicApi.Models;
using VetClinicApi.Repositories;

namespace VetClinicApi.Services
{
    public class UserService : IUserService
    {
        private readonly IUserRepository _userRepository;

        public UserService(IUserRepository userRepository)
        {
            _userRepository = userRepository;
        }

        public async Task<UserDto?> GetByIdAsync(int id)
        {
            var user = await _userRepository.GetByIdAsync(id);
            return user?.ToDto();
        }

        public async Task<UserDto?> GetByEmailAsync(string email)

```

```

        {
            var user = await
            _userRepository.GetByEmailAsync(email);
            return user?.ToDto();
        }
        public async Task CreateAsync(UserDto userDto)
        {
            var user = userDto.ToEntity();
            await _userRepository.AddAsync(user);
        }
        public async Task DeleteAsync(int id)
        {
            await _userRepository.DeleteAsync(id);
        }
    }
}

```

Листинг В.52 – Файл VetServiceService.cs

```

using VetClinicApi.DTOs;
using VetClinicApi.Mappers;
using VetClinicApi.Repositories;

namespace VetClinicApi.Services
{
    public class VetServiceService : IVetServiceService
    {
        private readonly IVetServiceRepository _repository;

        public VetServiceService(IVetServiceRepository repository)
        {
            _repository = repository;
        }
        public async Task<IEnumerable<VetServiceDto>>
        GetAllAsync()
        {
            var services = await _repository.GetAllAsync();
            return services.Select(VetServiceMapper.ToDto);
        }
        public async Task<VetServiceDto?> GetByIdAsync(int id)
        {
            var service = await _repository.GetByIdAsync(id);
            return VetServiceMapper.ToDto(service);
        }

        public async Task CreateAsync(VetServiceDto dto)
        {
            var service = VetServiceMapper.ToEntity(dto);
            await _repository.AddAsync(service);
        }
        public async Task DeleteAsync(int id)
        {
            await _repository.DeleteAsync(id);
        }
    }
}

```

Приложение Г

Исходный код мобильного приложения

(обязательное)


```

                                HorizontalOptions="End"/>
                            </VerticalStackLayout>
                        </Frame>
                    </DataTemplate>
                </CollectionView.ItemTemplate>
            </CollectionView>

            <!-- Прошедшие -->
            <Label Text="Прошедшие" FontAttributes="Bold"
FontSize="20" TextColor="DarkSlateBlue" Margin="0,10,0,0"/>

            <CollectionView ItemsSource="{Binding
PastAppointments}" SelectionMode="Single"

SelectionChanged="OnPastAppointmentSelected">
                <CollectionView.ItemTemplate>
                    <DataTemplate>
                        <Frame BackgroundColor="#f4f4f4"
BorderColor="#dddddd" CornerRadius="12" Margin="0,5" Padding="12">
                            <Grid Padding="12">
                                <Grid.GestureRecognizers>
                                    <TapGestureRecognizer
Command="{Binding BindingContext.OpenAppointmentDetailCommand,
Source={x:Reference appointmentsPage}}"
                                CommandParameter="{Binding .}" />
                                </Grid.GestureRecognizers>
                                <VerticalStackLayout Spacing="6">
                                    <Label Text="{Binding Date,
StringFormat='Дата: {0}'}" FontAttributes="Bold" FontSize="16"
TextColor="#222"/>
                                    <Label Text="{Binding Service,
StringFormat='Услуга: {0}'}" FontSize="14" TextColor="#333"/>
                                    <Label Text="{Binding Doctor,
StringFormat='Доктор: {0}'}" FontSize="14" TextColor="#555"/>
                                    <Label Text="{Binding Pet,
StringFormat='Питомец: {0}'}" FontSize="14" TextColor="#555"/>
                                </VerticalStackLayout>
                            </Grid>
                        </Frame>
                    </DataTemplate>
                </CollectionView.ItemTemplate>
            </CollectionView>
        </VerticalStackLayout>
    </ScrollView>
</ContentPage>

```

Листинг Г.2 – Файл AppointmentsPage.xaml.cs

```

using VetClinicAppMobile.ViewModels;
using System.Windows.Input;
namespace VetClinicAppMobile.Pages;

public partial class AppointmentsPage : ContentPage
{

```

```

        public ICommand OpenAppointmentDetailCommand { get; }
        private AppointmentsViewModel ViewModel => BindingContext as
AppointmentsViewModel;
        public AppointmentsPage()
        {
            InitializeComponent();
            OpenAppointmentDetailCommand = new
Command<AppointmentViewModel>(OpenAppointmentDetail);
            BindingContext = new AppointmentsViewModel();
        }
        protected override async void OnAppearing()
        {
            base.OnAppearing();
            if (ViewModel != null)
            {
                await ViewModel.LoadAppointments();
            }
        }
        private async void OpenAppointmentDetail(AppointmentViewModel
appointment)
        {
            if (appointment != null)
            {
                await Navigation.PushAsync(new
AppointmentDetailPage(appointment));
            }
        }
        private async void OnPastAppointmentSelected(object sender,
SelectionChangedEventArgs e)
        {
            if (e.CurrentSelection.FirstOrDefault() is
AppointmentViewModel selected)
            {
                await Navigation.PushAsync(new
AppointmentDetailPage(selected));
                ((CollectionView)sender).SelectedItem = null;
            }
        }
    }
}

```

Листинг Г.3 – Файл ApiService.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using VetClinicAppMobile.DTOs;

namespace VetClinicAppMobile.Services
{
    public class ApiService
    {
        private readonly HttpClient _httpClient;

        public ApiService()

```

```

    {
        _httpClient = new HttpClient
        {
            BaseAddress = new
Uri("http://192.168.0.29:5257/api/")
        };
    }
    private void AddAuthHeader()
    {
        if (Preferences.ContainsKey("auth_token"))
        {
            var token = Preferences.Get("auth_token",
string.Empty);
            Console.WriteLine("Token: " + token);
            if
(_httpClient.DefaultRequestHeaders.Contains("Authorization"))
            {
                _httpClient.DefaultRequestHeaders.Remove("Authorization");
            }

            _httpClient.DefaultRequestHeaders.Add("Authorization",
$"Bearer {token}");
        }
    }

    public async Task<List<VetServiceDto>>
GetVetServicesAsync()
    {
        AddAuthHeader();
        return await
_httpClient.GetFromJsonAsync<List<VetServiceDto>>("VetServices");
    }

    //Получение врачей по услуге
    public async Task<List<DoctorDto>>
GetDoctorsByServiceAsync(int serviceId)
    {
        AddAuthHeader();
        return await
_httpClient.GetFromJsonAsync<List<DoctorDto>>($"Doctors/by-
service/{serviceId}");
    }

    // Получение услуг врача
    public async Task<List<VetServiceDto>>
GetServicesByDoctorAsync(int doctorId)
    {
        AddAuthHeader();
        return await
_httpClient.GetFromJsonAsync<List<VetServiceDto>>($"DoctorVetServi
ce/doctor/{doctorId}");
    }

```

```

    }

    // Получение доступных дат для врача и услуги
    public async Task<List<DateTime>>
GetAvailableDatesAsync(int doctorId, int serviceId)
    {
        AddAuthHeader();
        return await
_httpClient.GetFromJsonAsync<List<DateTime>> (
            $"Appointments/available-
dates?doctorId={doctorId}&serviceId={serviceId}");
    }

    //Получение доступного времени по дате, врачу и услуге
    public async Task<List<TimeSpan>>
GetAvailableTimesAsync(int doctorId, int serviceId, DateTime date)
    {
        AddAuthHeader();
        string formattedDate = date.ToString("yyyy-MM-dd");
        return await
_httpClient.GetFromJsonAsync<List<TimeSpan>> (
            $"Appointments/available-
times?doctorId={doctorId}&serviceId={serviceId}&date={formattedDate}");
    }

    // Получение питомцев клиента
    public async Task<List<PetDto>> GetClientPetsAsync(int
clientId)
    {
        AddAuthHeader();
        return await
_httpClient.GetFromJsonAsync<List<PetDto>>($"Pets/client/{clientId}");
    }

    // Запись на прием
    public async Task<bool>
BookAppointmentAsync(BookAppointmentDto dto)
    {
        AddAuthHeader();
        var json = JsonSerializer.Serialize(dto);
        Console.WriteLine("Отправляемый DTO:");
        Console.WriteLine(json);
        var content = new StringContent(json, Encoding.UTF8,
"application/json");

        var response = await
_httpClient.PostAsync("Appointments/book", content);
        Console.WriteLine("Статус-код ответа: " +
response.StatusCode);
        var responseBody = await
response.Content.ReadAsStringAsync();
        Console.WriteLine("Ответ сервера: " + responseBody);
        return response.IsSuccessStatusCode;
    }

```

```

    }

    //Получение текущего пользователя
    public Task<int> GetCurrentUserIdAsync()
    {
        AddAuthHeader();
        if (Preferences.ContainsKey("ClientId"))
            return Task.FromResult(Preferences.Get("ClientId", 0));
        return Task.FromResult(0);
    }
}
}

```

Листинг Г.4 – Файл AppointmentsViewModel.cs

```

using System.Net.Http.Json;
using VetClinicAppMobile.DTOs;

namespace VetClinicAppMobile.ViewModels
{
    public class AppointmentsViewModel
    {
        public ObservableCollection<AppointmentViewModel>
UpcomingAppointments { get; set; } = new();
        public ObservableCollection<AppointmentViewModel>
PastAppointments { get; set; } = new();
        public ICommand CancelCommand { get; set; }

        public AppointmentsViewModel()
        {
            CancelCommand = new Command<int>(async (id) => await
CancelAppointment(id));
            LoadAppointments();
        }
        public async Task LoadAppointments()
        {
            try
            {
                var clientId = Preferences.Get("ClientId", 0);
                var http = new HttpClient();
                var appointments = await
http.GetFromJsonAsync<List<AppointmentDto>>(
$"http://192.168.0.29:5257/api/appointments/client/{clientId}");

                UpcomingAppointments.Clear();
                PastAppointments.Clear();

                foreach (var app in appointments.Where(a =>
a.ClientId == clientId))
                {
                    var vm = new AppointmentViewModel
                    {

```

```

        Id = app.Id,
        Date =
app.AvailableDate.ToString("dd.MM.yyyy") ?? "не указана",
        Time =
app.AvailableTime.ToString("hh':'mm"),
        Service = app.ServicName,
        Doctor = app.LastName + " " +
app.FirstName,
        Pet = app.PetsName,
        Recommendation = app.Recommendation,
        IsUpcoming = app.AppointmentStatus ==
"Запланировано"
    };

    if (vm.IsUpcoming)
        UpcomingAppointments.Add(vm);
    else
        PastAppointments.Add(vm);
    }
}
catch (Exception ex)
{
    await
Application.Current.MainPage.DisplayAlert("Ошибка", ex.Message,
"OK");
}
}
private async Task CancelAppointment(int id)
{
    var appointment =
UpcomingAppointments.FirstOrDefault(a => a.Id == id);

    if (appointment == null) return;

    // Запрос подтверждения
    bool answer = await
Application.Current.MainPage.DisplayAlert(
        "Подтверждение отмены",
        $"Вы действительно хотите отменить запись на
{appointment.Date} в {appointment.Time}?",
        "Да", "Нет");

    if (!answer) return;

    try
    {
        var http = new HttpClient();
        var response = await
http.DeleteAsync($"http://192.168.0.29:5257/api/appointments/{id}")
);

        if (response.IsSuccessStatusCode)
        {

```

```

        // Удаляем из списка и обновляем
        UpcomingAppointments.Remove(appointment);
        await
Application.Current.MainPage.DisplayAlert("Успех", "Запись успешно
отменена", "OK");
    }
    else
    {
        var error = await
response.Content.ReadAsStringAsync();
        await
Application.Current.MainPage.DisplayAlert("Ошибка", error, "OK");
    }
}
catch (Exception ex)
{
    await
Application.Current.MainPage.DisplayAlert("Ошибка", ex.Message,
"OK");
}
}
}
}

```

Листинг Г.5 – Файл AppointmentDetailPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="VetClinicAppMobile.Pages.AppointmentDetailPage"
    Title="Прием">
    <ScrollView>
        <VerticalStackLayout Padding="20" Spacing="20">

            <!-- Дата -->
            <Label Text="{Binding Date}"
                FontSize="22"
                FontAttributes="Bold"
                TextColor="DarkSlateBlue" />

            <!-- Услуга -->
            <Frame BackgroundColor="#f2f2f2" Padding="10"
                CornerRadius="12">
                <VerticalStackLayout Spacing="4">
                    <Label Text="Услуга:" FontAttributes="Bold"
                        TextColor="#444" />
                    <Label Text="{Binding Service}"
                        TextColor="#222"/>
                </VerticalStackLayout>
            </Frame>

```

```

        <!-- Врач -->
        <Frame BackgroundColor="#f2f2f2" Padding="10"
CornerRadius="12">
            <VerticalStackLayout Spacing="4">
                <Label Text="Врач:" FontAttributes="Bold"
TextColor="#444" />
                <Label Text="{Binding Doctor}"
TextColor="#222"/>
            </VerticalStackLayout>
        </Frame>

        <!-- Питомец -->
        <Frame BackgroundColor="#f2f2f2" Padding="10"
CornerRadius="12">
            <VerticalStackLayout Spacing="4">
                <Label Text="Питомец:" FontAttributes="Bold"
TextColor="#444" />
                <Label Text="{Binding Pet}" TextColor="#222"/>
            </VerticalStackLayout>
        </Frame>

        <!-- Рекомендации -->
        <Frame BackgroundColor="#e6f7ff" Padding="15"
CornerRadius="12" BorderColor="#007ACC">
            <VerticalStackLayout Spacing="6">
                <Label Text="Рекомендации врача" FontSize="18"
FontAttributes="Bold" TextColor="#005A9E" />
                <Label Text="{Binding Recommendation}"
TextColor="#003366" LineBreakMode="WordWrap"/>
            </VerticalStackLayout>
        </Frame>
    </VerticalStackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.6 – Файл AppointmentDetailPage.xaml.cs

```

using VetClinicAppMobile.Models;
namespace VetClinicAppMobile.Pages;

public partial class AppointmentDetailPage : ContentPage
{
    public AppointmentDetailPage(AppointmentViewModel model)
    {
        InitializeComponent();
        BindingContext = model;
    }
}

```

Листинг Г.7 – Файл BookingPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

```

```

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
      x:Class="VetClinicAppMobile.Pages.BookingPage"
      Title="Запись">
    <ScrollView>
      <VerticalStackLayout Padding="20" Spacing="20">

        <Label Text="Запись на приём"
          FontSize="22"
          FontAttributes="Bold"
          TextColor="DarkSlateBlue" />
        <!-- Услуга -->
        <Frame BackgroundColor="#f2f2f2" Padding="10"
CornerRadius="12">
          <VerticalStackLayout Spacing="4">
            <Label Text="Услуга" FontAttributes="Bold"
TextColor="#444" />
            <Picker x:Name="ServicePicker"
              Title="Выберите услугу"

SelectedIndexChanged="OnServiceSelected"
              TextColor="#222"/>
          </VerticalStackLayout>
        </Frame>
        <!-- Врач -->
        <Frame BackgroundColor="#f2f2f2" Padding="10"
CornerRadius="12">
          <VerticalStackLayout Spacing="4">
            <Label Text="Врач" FontAttributes="Bold"
TextColor="#444" />
            <Picker x:Name="DoctorPicker"
              Title="Выберите врача"

SelectedIndexChanged="OnDoctorSelected"
              TextColor="#222"/>
          </VerticalStackLayout>
        </Frame>
        <!-- Дата -->
        <Frame BackgroundColor="#f2f2f2" Padding="10"
CornerRadius="12">
          <VerticalStackLayout Spacing="4">
            <Label Text="Дата" FontAttributes="Bold"
TextColor="#444" />
            <Picker x:Name="DatePicker"
              Title="Выберите дату"
              SelectedIndexChanged="OnDateSelected"
              TextColor="#222"/>
          </VerticalStackLayout>
        </Frame>
        <!-- Время -->
        <Frame BackgroundColor="#f2f2f2" Padding="10"
CornerRadius="12">
          <VerticalStackLayout Spacing="4">

```

```

        <Label Text="Время" FontAttributes="Bold"
TextColor="#444" />
        <Picker x:Name="TimePicker"
            Title="Выберите время"
            TextColor="#222"/>
    </VerticalStackLayout>
</Frame>
<!-- Питомец -->
<Frame BackgroundColor="#f2f2f2" Padding="10"
CornerRadius="12">
    <VerticalStackLayout Spacing="4">
        <Label Text="Питомец" FontAttributes="Bold"
TextColor="#444" />
        <Picker x:Name="PetPicker"
            Title="Выберите питомца"
            TextColor="#222"/>
    </VerticalStackLayout>
</Frame>
<!-- Кнопка -->
<Button Text="Записаться"
    Clicked="OnBookClicked"
    BackgroundColor="#007ACC"
    TextColor="White"
    CornerRadius="12"
    FontAttributes="Bold"
    Padding="12,8"
    Margin="0,10,0,0"/>

    </VerticalStackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.8 – Файл BookingPage.xaml.cs

```

using VetClinicAppMobile.DTOs;
using VetClinicAppMobile.Services;

namespace VetClinicAppMobile.Pages;

public partial class BookingPage : ContentPage
{
    private readonly ApiService _api = new();
    private int _userId;
    private VetServiceDto _preselectedService;

    public BookingPage()
    {
        InitializeComponent();
        InitAsync();
    }

    public BookingPage(VetServiceDto selectedService) : this()
    {
        _preselectedService = selectedService;
    }

```

```

    }

    private async void InitAsync()
    {
        _userId = await _api.GetCurrentUserIdAsync(); //
        сначала получаем ID
        await LoadServices();
        await LoadPets();

        if (_preselectedService != null)
        {
            ServicePicker.SelectedItem =
ServicePicker.ItemsSource
                .Cast<VetServiceDto>()
                .FirstOrDefault(s => s.Id ==
_preselectedService.Id);
        }
    }

    private async Task LoadServices()
    {
        var services = await _api.GetVetServicesAsync();
        ServicePicker.ItemsSource = services;
        //ServicePicker.SelectedItem = null;
        ServicePicker.ItemDisplayBinding = new
Binding("ServiceName");
    }

    private async Task LoadPets()
    {
        var pets = await _api.GetClientPetsAsync(_userId);
        PetPicker.ItemsSource = pets;
        PetPicker.SelectedItem = null;
        PetPicker.ItemDisplayBinding = new Binding("PetsName");
    }

    private async void OnServiceSelected(object sender, EventArgs
e)
    {
        if (ServicePicker.SelectedItem is not VetServiceDto
service)
        {
            DoctorPicker.ItemsSource = null;
            DatePicker.ItemsSource = null;
            TimePicker.ItemsSource = null;
            return;
        }

        var doctors = await
_api.GetDoctorsByServiceAsync(service.Id);
        DoctorPicker.ItemsSource = doctors;
        DoctorPicker.SelectedItem = null;
        DoctorPicker.ItemDisplayBinding = new Binding(".");
    }

```

```

        // DoctorPicker.ItemDisplayBinding = new
Binding("LastName");

        DatePicker.ItemsSource = null;
        TimePicker.ItemsSource = null;
    }

    private async void OnDoctorSelected(object sender, EventArgs
e)
    {
        if (DoctorPicker.SelectedItem is not DoctorDto doctor ||
ServicePicker.SelectedItem is not VetServiceDto
service)
        {
            DatePicker.ItemsSource = null;
            TimePicker.ItemsSource = null;
            return;
        }

        var dates = await _api.GetAvailableDatesAsync(doctor.Id,
service.Id);
        if (dates == null || dates.Count == 0)
        {
            DatePicker.ItemsSource = new List<string> { "Нет
доступных дат" };
            DatePicker.SelectedItem = "Нет доступных дат";
            TimePicker.ItemsSource = null;
            return;
        }

        var dateStrings = dates.Select(d => d.ToString("yyyy-MM-
dd")).ToList();
        DatePicker.ItemsSource = dateStrings;
        DatePicker.SelectedItem = null;

        TimePicker.ItemsSource = null;
    }

    private async void OnDateSelected(object sender, EventArgs e)
    {
        if (DatePicker.SelectedItem is not string dateString ||
ServicePicker.SelectedItem is not VetServiceDto
service ||
DoctorPicker.SelectedItem is not DoctorDto doctor)
        {
            TimePicker.ItemsSource = null;
            return;
        }

        if (!DateTime.TryParse(dateString, out var selectedDate))
        {
            TimePicker.ItemsSource = null;
            return;
        }
    }

```

```

    }

    var times = await _api.GetAvailableTimesAsync(doctor.Id,
service.Id, selectedDate);
    if (times == null || times.Count == 0)
    {
        TimePicker.ItemsSource = new List<string> { "Нет
доступного времени" };
        TimePicker.SelectedItem = "Нет доступного времени";
        return;
    }

    var timeStrings = times.Select(t =>
t.ToString(@"hh\:mm")).ToList();
    TimePicker.ItemsSource = timeStrings;
    TimePicker.SelectedItem = null;
}

private async void OnBookClicked(object sender, EventArgs e)
{
    if (ServicePicker.SelectedItem is not VetServiceDto
service ||
        DoctorPicker.SelectedItem is not DoctorDto doctor ||
        DatePicker.SelectedItem is not string dateString ||
        TimePicker.SelectedItem is not string timeString ||
        PetPicker.SelectedItem is not PetDto pet ||
        dateString == "Нет доступных дат" || timeString ==
"Нет доступного времени")
    {
        await DisplayAlert("Ошибка", "Пожалуйста, заполните
все поля", "OK");
        return;
    }

    if (!DateTime.TryParse(dateString, out var date) ||
        !TimeSpan.TryParse(timeString, out var time))
    {
        await DisplayAlert("Ошибка", "Некорректная дата или
время", "OK");
        return;
    }

    var dto = new BookAppointmentDto
    {
        ServiceId = service.Id,
        DoctorId = doctor.Id,
        ClientId = Preferences.Get("ClientId", 0),
        AvailableDate = date,
        AvailableTime = time,
        PetId = pet.Id,
    };

    bool success = await _api.BookAppointmentAsync(dto);

```

```

        if (success)
        {
            await DisplayAlert("Успех", "Вы успешно записались на
прием", "OK");
            await Shell.Current.GoToAsync("//AppointmentsPage");
        }
        else
        {
            await DisplayAlert("Ошибка", "Не удалось записаться.
Попробуйте позже.", "OK");
        }
    }
}

```

Листинг Г.9 – Файл DoctorHomePage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="VetClinicAppMobile.Pages.DoctorHomePage"
            Title="Мои приемы">
    <VerticalStackLayout Padding="10">
        <Label Text="Запланированные приёмы"
            FontSize="20"
            FontAttributes="Bold"
            Margin="0,10,0,5"/>
        <CollectionView x:Name="UpcomingAppointmentsView">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame BorderColor="LightGray"
CornerRadius="10" Padding="10" Margin="0,5">
                        <VerticalStackLayout>
                            <Label Text="{Binding ClientName}"
FontSize="16" FontAttributes="Bold"/>
                            <Label Text="{Binding PetsName}" />
                            <Label Text="{Binding
DateTimeDisplay}" />
                            <Button Text="Подробнее"
                                Clicked="OnAppointmentTapped"
                                CommandParameter="{Binding Id}" />
                        </VerticalStackLayout>
                    </Frame>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>

        <Label Text="Прошедшие приёмы"
            FontSize="20"
            FontAttributes="Bold"
            Margin="0,20,0,5"/>
        <CollectionView x:Name="PastAppointmentsView">
            <CollectionView.ItemTemplate>
                <DataTemplate>

```

```

        <Frame BorderColor="LightGray"
        CornerRadius="10" Padding="10" Margin="0,5">
            <VerticalStackLayout>
                <Label Text="{Binding ClientName}"
                FontSize="16" FontAttributes="Bold"/>
                <Label Text="{Binding PetsName}" />
                <Label Text="{Binding
DateTimeDisplay}" />
                <Button Text="Подробнее"
                Clicked="OnAppointmentTapped"
                CommandParameter="{Binding Id}" />
            </VerticalStackLayout>
        </Frame>
    </DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</VerticalStackLayout>
</ContentPage>

```

Листинг Г.10 – Файл DoctorHomePage.xaml.cs

```

using VetClinicAppMobile.Models;
using System.Collections.ObjectModel;

namespace VetClinicAppMobile.Pages;

public partial class DoctorHomePage : ContentPage
{
    private readonly IDoctorAppointmentApiService _appointmentApi;

    public ObservableCollection<AppointmentViewModel>
UpcomingAppointments { get; set; } = new();
    public ObservableCollection<AppointmentViewModel>
PastAppointments { get; set; } = new();
    public DoctorHomePage(IDoctorAppointmentApiService
appointmentApi)
    {
        InitializeComponent();
        _appointmentApi = appointmentApi;
        BindingContext = this;
    }

    protected override async void OnAppearing()
    {
        base.OnAppearing();

        var doctorId = Preferences.Get("DoctorId", 0);
        if (doctorId == 0)
        {
            await DisplayAlert("Ошибка", "Не удалось получить ID
доктора", "OK");
            return;
        }
    }
}

```

```

        var appointments = await
            _appointmentApi.GetAppointmentsByDoctorIdAsync(doctorId);

        var now = DateTime.Now;

        var upcoming = appointments
            .Where(a => a.AppointmentStatus == "Запланирован" &&
                a.DateTime.HasValue && a.DateTime > now)
            .OrderBy(a => a.DateTime)
            .Select(DoctorAppointmentViewModel.FromDto);

        var past = appointments
            .Where(a => a.AppointmentStatus == "Завершён" &&
                a.DateTime.HasValue && a.DateTime <= now)
            .OrderByDescending(a => a.DateTime)
            .Select(DoctorAppointmentViewModel.FromDto);

        UpcomingAppointments.Clear();
        PastAppointments.Clear();

        foreach (var app in upcoming)
            UpcomingAppointments.Add(app);

        foreach (var app in past)
            PastAppointments.Add(app);

        UpcomingAppointmentsView.ItemsSource =
            UpcomingAppointments;
        PastAppointmentsView.ItemsSource = PastAppointments;
    }
    private async void OnAppointmentTapped(object sender,
        EventArgs e)
    {
        var button = sender as Button;
        if (button?.CommandParameter is int appointmentId)
        {
            await Navigation.PushAsync(new
                MedicalRecordsPage(appointmentId));
        }
    }
}

```

Листинг Г.11 – Файл DoctorSchedulePage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class="VetClinicAppMobile.Pages.DoctorSchedulePage"
        Title="Моё расписание">
    <ScrollView>
        <VerticalStackLayout Padding="20" Spacing="20">

            <!-- Заголовок -->

```

```

        <Label Text="Выберите дни недели:"
FontAttributes="Bold" FontSize="20" TextColor="DarkSlateBlue" />

        <!-- Чекбоксы -->
        <VerticalStackLayout Spacing="8">
            <HorizontalStackLayout>
                <CheckBox x:Name="MondayCheck" />
                <Label Text="Понедельник"
VerticalOptions="Center" FontSize="16" />
            </HorizontalStackLayout>
            <HorizontalStackLayout>
                <CheckBox x:Name="TuesdayCheck" />
                <Label Text="Вторник" VerticalOptions="Center"
FontSize="16" />
            </HorizontalStackLayout>
            <HorizontalStackLayout>
                <CheckBox x:Name="WednesdayCheck" />
                <Label Text="Среда" VerticalOptions="Center"
FontSize="16" />
            </HorizontalStackLayout>
            <HorizontalStackLayout>
                <CheckBox x:Name="ThursdayCheck" />
                <Label Text="Четверг" VerticalOptions="Center"
FontSize="16" />
            </HorizontalStackLayout>
            <HorizontalStackLayout>
                <CheckBox x:Name="FridayCheck" />
                <Label Text="Пятница" VerticalOptions="Center"
FontSize="16" />
            </HorizontalStackLayout>
        </VerticalStackLayout>

        <!-- Время и длительность -->
        <Label Text="С:" FontSize="16" />
        <TimePicker x:Name="StartTimePicker" Time="09:00" />

        <Label Text="До:" FontSize="16" />
        <TimePicker x:Name="EndTimePicker" Time="17:00" />

        <Label Text="Длительность слота (мин):" FontSize="16"
/>
        <Entry x:Name="SlotDurationEntry" Keyboard="Numeric"
Text="30" />

        <!-- Кнопка -->
        <Button Text="Добавить расписание"
Clicked="OnAddScheduleClicked"
BackgroundColor="#4a90e2"
TextColor="White"
CornerRadius="10"
FontSize="16"
HeightRequest="45" />

```

```

        <!-- Слоты -->
        <Label Text="Добавленные слоты:" FontAttributes="Bold"
FontSize="20" TextColor="DarkSlateBlue" Margin="0,10,0,0"/>

        <CollectionView x:Name="SlotsCollectionView"
SelectionMode="None">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame BackgroundColor="#f0f8ff"
                        BorderColor="#d0e7ff"
                        CornerRadius="12"
                        Margin="0,5"
                        Padding="12"
                        HasShadow="True">
                        <VerticalStackLayout Spacing="6">
                            <Label Text="{Binding Date}"
FontSize="16" TextColor="#333" />
                            <Label Text="{Binding Time}"
FontSize="14" TextColor="#555" />
                        </VerticalStackLayout>
                    </Frame>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </VerticalStackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.12 – Файл DoctorSchedulePage.xaml.cs

```

using System.Collections.ObjectModel;
using Newtonsoft.Json;

namespace VetClinicAppMobile.Pages;

public partial class DoctorSchedulePage : ContentPage
{
    private readonly HttpClient _httpClient = new();
    public ObservableCollection<ScheduleSlot> Slots { get; set; }
    = new();

    public DoctorSchedulePage()
    {
        InitializeComponent();
        SlotsCollectionView.ItemsSource = Slots;
        LoadScheduleFromServer();
    }
    private async void LoadScheduleFromServer()
    {
        try
        {
            var response = await
            _httpClient.GetAsync($"/api/schedule/{_doctorId}");
            response.EnsureSuccessStatusCode();

```

```

        var serverSlots =
JsonConvert.DeserializeObject<List<DoctorScheduleDto>>(await
response.Content.ReadAsStringAsync());

        Slots.Clear();
        foreach (var slot in serverSlots)
        {
            Slots.Add(new ScheduleSlot
            {
                Date =
                $"{ToRussianDayOfWeek(slot.Date.DayOfWeek)},
                {slot.Date:dd.MM.yyyy}",
                Time = $"{slot.StartTime:hh\\\:mm} -
                {slot.EndTime:hh\\\:mm}"
            });
        }
    }
    catch (Exception ex)
    {
        await DisplayAlert("Ошибка", $"Не удалось загрузить
расписание: {ex.Message}", "OK");
    }
}

private async void OnAddScheduleClicked(object sender,
EventArgs e)
{
    var selectedDays = new List<DayOfWeek>();
    if (MondayCheck.IsChecked)
selectedDays.Add(DayOfWeek.Monday);
    if (TuesdayCheck.IsChecked)
selectedDays.Add(DayOfWeek.Tuesday);
    if (WednesdayCheck.IsChecked)
selectedDays.Add(DayOfWeek.Wednesday);
    if (ThursdayCheck.IsChecked)
selectedDays.Add(DayOfWeek.Thursday);
    if (FridayCheck.IsChecked)
selectedDays.Add(DayOfWeek.Friday);

    if (!int.TryParse(SlotDurationEntry.Text, out int
slotMinutes) || slotMinutes <= 0)
    {
        await DisplayAlert("Ошибка", "Введите корректную
длительность слота", "OK");
        return;
    }

    var start = StartTimePicker.Time;
    var end = EndTimePicker.Time;

    if (end <= start)
    {

```

```

        await DisplayAlert("Ошибка", "Время окончания должно
        быть позже начала", "OK");
        return;
    }

    var today = DateTime.Today;
    var newSlots = new List<DoctorScheduleDto>();

    foreach (var day in selectedDays)
    {
        var targetDate = GetNextWeekday(today, day);
        var current = start;
        while (current + TimeSpan.FromMinutes(slotMinutes) <=
end)
        {
            var dto = new DoctorScheduleDto
            {
                DoctorId = _doctorId,
                Date = targetDate,
                StartTime = current,
                EndTime = current +
TimeSpan.FromMinutes(slotMinutes)
            };

            newSlots.Add(dto);
            current += TimeSpan.FromMinutes(slotMinutes);
        }
    }
    try
    {
        var response = await
_httpClient.PostAsJsonAsync("url/api/schedule", newSlots);
        response.EnsureSuccessStatusCode();
        await DisplayAlert("Успех", "Расписание успешно
добавлено", "OK");
        LoadScheduleFromServer();
    }
    catch (Exception ex)
    {
        await DisplayAlert("Ошибка", $"Не удалось сохранить
расписание: {ex.Message}", "OK");
    }
}

private DateTime GetNextWeekday(DateTime start, DayOfWeek
target)
{
    int daysToAdd = ((int)target - (int)start.DayOfWeek + 7) %
7;
    return start.AddDays(daysToAdd == 0 ? 7 : daysToAdd);
}

private string ToRussianDayOfWeek(DayOfWeek day) => day switch

```

```

{
    DayOfWeek.Monday => "Понедельник",
    DayOfWeek.Tuesday => "Вторник",
    DayOfWeek.Wednesday => "Среда",
    DayOfWeek.Thursday => "Четверг",
    DayOfWeek.Friday => "Пятница",
    DayOfWeek.Saturday => "Суббота",
    DayOfWeek.Sunday => "Воскресенье",
    _ => ""
};
}

```

Листинг Г.13 – Файл LoginPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="VetClinicAppMobile.Pages.LoginPage"
            Title="Войти в систему">
    <VerticalStackLayout Padding="30" Spacing="20"
VerticalOptions="Center">
        <Entry x:Name="LoginEntry"
            Placeholder="Login"
            Keyboard="Login"
            Text=""
            FontSize="18" />

        <Entry x:Name="PasswordEntry"
            Placeholder="Пароль"
            IsPassword="True"
            Text=""
            FontSize="18" />

        <Button Text="Вход"
            FontSize="20"
            Clicked="OnLoginClicked" />

        <Label Text="Еще не зарегистрировались?"
            FontSize="14"
            HorizontalOptions="Center" />

        <Button Text="Регистрация"
            FontSize="18"
            Clicked="OnRegisterClicked"
            HorizontalOptions="Center" />

    </VerticalStackLayout>
</ContentPage>

```

Листинг Г.14 – Файл LoginPage.xaml.cs

```

namespace VetClinicAppMobile.Pages;

```



```

        Preferences.Set("DoctorId",
loginResponse.DoctorId.Value);
        if (loginResponse.AdminId.HasValue)
            Preferences.Set("AdminId",
loginResponse.AdminId.Value);

        await DisplayAlert("Успешно", $"Вход выполнен
как {loginResponse.Role}", "OK");
        // Переход на нужную страницу по роли
        if (loginResponse.Role == "Client")
        {
            (Shell.Current as
AppShell)?.SetClientTabsVisible(true);
            (Shell.Current as
AppShell)?.SetDoctorTabsVisible(false);
            await
Shell.Current.GoToAsync("//AppointmentsPage");
        }
        else if (loginResponse.Role == "Doctor")
        {
            (Shell.Current as
AppShell)?.SetClientTabsVisible(false);
            (Shell.Current as
AppShell)?.SetDoctorTabsVisible(true);
            await
Shell.Current.GoToAsync("//DoctorHomePage");
        }
        else if (loginResponse.Role == "Admin")
        {
            await
Shell.Current.GoToAsync("//AdminDashboard");
        }
    }
    else
    {
        var error = await
response.Content.ReadAsStringAsync();
        Console.WriteLine($"StatusCode:
{(int)response.StatusCode}, Body: {error}");
        await DisplayAlert("Ошибка входа", error, "OK");
    }
}
catch (Exception ex)
{
    await DisplayAlert("Ошибка", $"Сетевая ошибка:
{ex.Message}", "OK");
}
}

private async void OnRegisterClicked(object sender, EventArgs e)
{

```

```

        await Navigation.PushAsync(new RegisterPage());
    }
}

```

Листинг Г.15 – Файл PetEditPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class="VetClinicAppMobile.Pages.PetEditPage"
        Title="Питомец">
    <ScrollView>
        <VerticalStackLayout Padding="20" Spacing="15">
            <Label Text="Информация о питомце"
                FontSize="24"
                FontAttributes="Bold"
                TextColor="#333" />

            <Frame BackgroundColor="White" CornerRadius="12"
                Padding="15" HasShadow="True">
                <VerticalStackLayout Spacing="10">
                    <Entry Placeholder="Кличка"
                        Text="{Binding Pet.PetsName}"
                        FontSize="16"
                        TextColor="#000" />

                    <Entry Placeholder="Тип"
                        Text="{Binding Pet.PetsType}"
                        FontSize="16"
                        TextColor="#000" />

                    <Entry Placeholder="Порода"
                        Text="{Binding Pet.Breed}"
                        FontSize="16"
                        TextColor="#000" />

                    <Picker Title="Пол"
                        SelectedItem="{Binding Pet.Gender}"
                        FontSize="16"
                        TextColor="#000">
                        <Picker.ItemsSource>
                            <x:Array Type="{x:Type x:String}">
                                <x:String>Мужской</x:String>
                                <x:String>Женский</x:String>
                            </x:Array>
                        </Picker.ItemsSource>
                    </Picker>

                    <DatePicker Date="{Binding Pet.BirthDate}"
                        FontSize="16"
                        TextColor="#000" />
                </VerticalStackLayout>
            </Frame>

```

```

        <Button Text="Сохранить"
                Command="{Binding SaveCommand}"
                BackgroundColor="#2196F3"
                TextColor="White"
                FontAttributes="Bold"
                CornerRadius="10"
                Padding="12" />
    </VerticalStackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.16 – Файл PetEditPage.xaml.cs

```

using VetClinicAppMobile.ViewModels;
using VetClinicAppMobile.DTOs;

namespace VetClinicAppMobile.Pages;

public partial class PetEditPage : ContentPage
{
    public PetEditPage(PetDto? pet, IPetService petService,
        Action onSave)
    {
        InitializeComponent();
        BindingContext = new PetEditViewModel(pet, petService,
            onSave);
    }
}

```

Листинг Г.17 – Файл PetEditViewModel.cs

```

using System.Windows.Input;
using VetClinicAppMobile.DTOs;
using VetClinicAppMobile.Services;

namespace VetClinicAppMobile.ViewModels
{
    public class PetEditViewModel
    {
        public PetDto Pet { get; set; }
        private readonly IPetService _petService;
        private readonly Action _onSaved;
        public ICommand SaveCommand { get; }
        public PetEditViewModel(PetDto? pet, IPetService
            petService, Action onSave)
        {
            Pet = pet ?? new PetDto
            {
                ClientId = Preferences.Get("ClientId", 0),
                BirthDate = DateTime.Now };

            _petService = petService;
            _onSaved = onSave;
        }
    }
}

```

```

        SaveCommand = new Command(async () => await
SaveAsync());
    }
    private async Task SaveAsync()
    {
        if (Pet.Id == 0)
            await _petService.CreateAsync(Pet);
        else
            await _petService.UpdateAsync(Pet.Id, Pet);

        _onSaved?.Invoke();
    }
}

```

Листинг Г.18 – Файл PetService.cs

```

using VetClinicAppMobile.DTOs;

namespace VetClinicAppMobile.Services
{
    public interface IPetService
    {
        Task CreateAsync(PetDto pet);
        Task UpdateAsync(int id, PetDto pet);
    }
    public class PetService : IPetService
    {
        private readonly HttpClient _http;

        public PetService(HttpClient http)
        {
            _http = http;
        }

        public async Task CreateAsync(PetDto pet)
        {
            await _http.PostAsJsonAsync("api/pets", pet);
        }

        public async Task UpdateAsync(int id, PetDto pet)
        {
            await _http.PutAsJsonAsync($"api/pets/{id}", pet);
        }
    }
}

```

Листинг Г.19 – Файл ProfilePage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

```

```

x:Class="VetClinicAppMobile.Pages.ProfilePage"
Title="Профиль">

<ScrollView>
    <VerticalStackLayout Padding="20" Spacing="20">
        <Label Text="Профиль клиента"
            FontSize="28"
            FontAttributes="Bold"
            TextColor="#333" />

        <Frame BackgroundColor="White" CornerRadius="12"
            Padding="15" HasShadow="True">
            <VerticalStackLayout Spacing="10">
                <Label Text="{Binding FullName}" FontSize="20"
                    FontAttributes="Bold" TextColor="#444" />
                <Label Text="{Binding PhoneNumber}"
                    FontSize="16" TextColor="#666" />
                <Label Text="{Binding Email}" FontSize="16"
                    TextColor="#666" />
            </VerticalStackLayout>
        </Frame>

        <Label Text="Питомцы:"
            FontSize="22"
            FontAttributes="Bold"
            TextColor="#333" />

        <CollectionView ItemsSource="{Binding Pets}"
            SelectionMode="Single"
            SelectionChanged="OnPetSelected">
            <CollectionView.ItemsLayout>
                <LinearItemsLayout Orientation="Vertical" />
            </CollectionView.ItemsLayout>
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame BackgroundColor="White"
                        BorderColor="#DDD"
                        CornerRadius="15"
                        Padding="15"
                        Margin="5"
                        HasShadow="True">
                        <VerticalStackLayout Spacing="5">
                            <Label Text="{Binding PetsName}"
                                FontSize="18" FontAttributes="Bold" TextColor="#222" />
                            <Label Text="{Binding PetsType}"
                                FontSize="16" TextColor="#555" />
                            <Label Text="{Binding Breed}"
                                FontSize="16" TextColor="#555" />
                            <Label Text="{Binding Gender,
                                StringFormat='Пол: {0}'}" FontSize="14" TextColor="#777" />
                            <Label Text="{Binding BirthDate,
                                StringFormat='Дата рождения: {0:dd.MM.yyyy}'}" FontSize="14"
                                TextColor="#777" />
                        </VerticalStackLayout>
                    </Frame>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </VerticalStackLayout>
</ScrollView>

```

```

        </VerticalStackLayout>
    </Frame>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>

<Button Text="Добавить питомца"
        Clicked="OnAddPetClicked"
        BackgroundColor="#4CAF50"
        TextColor="White"
        CornerRadius="10"
        FontAttributes="Bold"
        Padding="12" />

<Button Text="Выйти из аккаунта"
        Clicked="OnLogoutClicked"
        BackgroundColor="#F44336"
        TextColor="White"
        CornerRadius="10"
        FontAttributes="Bold"
        Padding="12" />
</VerticalStackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.20 – Файл RegisterPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class="VetClinicAppMobile.Pages.RegisterPage"
        Title="Регистрация">
    <ScrollView>
        <VerticalStackLayout Padding="30" Spacing="15"
VerticalOptions="Center">

            <Entry x:Name="LastNameEntry"
                Placeholder="Фамилия"
                FontSize="18" />

            <Entry x:Name="FirstNameEntry"
                Placeholder="Имя"
                FontSize="18" />

            <Entry x:Name="MiddleNameEntry"
                Placeholder="Отчество"
                FontSize="18" />

            <Entry x:Name="PhoneEntry"
                Placeholder="Номер телефона"
                Keyboard="Telephone"
                FontSize="18" />

```

```

        <Entry x:Name="EmailEntry"
            Placeholder="Email"
            Keyboard="Email"
            FontSize="18" />

        <Entry x:Name="LoginEntry"
            Placeholder="Login"
            Keyboard="Login"
            FontSize="18" />

        <Entry x:Name="PasswordEntry"
            Placeholder="Пароль"
            IsPassword="True"
            FontSize="18" />

        <Button Text="Регистрация"
            FontSize="20"
            Clicked="OnRegisterClicked" />

        <Label Text="Уже зарегистрировались?"
            FontSize="14"
            HorizontalOptions="Center" />

        <Button Text="Вход"
            FontSize="18"
            Clicked="OnLoginClicked"
            HorizontalOptions="Center" />
    </VerticalStackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.21 – Файл RegisterPage.xaml.cs

```

using VetClinicAppMobile.DTOs;
using System.Net.Http;

namespace VetClinicAppMobile.Pages;

public partial class RegisterPage : ContentPage
{
    public RegisterPage()
    {
        InitializeComponent();
    }

    private async void OnRegisterClicked(object sender, EventArgs
e)
    {
        string lastName = LastNameEntry.Text?.Trim();
        string firstName = FirstNameEntry.Text?.Trim();
        string middleName = MiddleNameEntry.Text?.Trim();
        string phone = PhoneEntry.Text?.Trim();
        string email = EmailEntry.Text?.Trim();
        string password = PasswordEntry.Text;
    }
}

```

```

        if (string.IsNullOrEmpty(lastName) ||
            string.IsNullOrEmpty(firstName) ||
            string.IsNullOrEmpty(phone) ||
            string.IsNullOrEmpty(email) ||
            string.IsNullOrEmpty(password))
        {
            await DisplayAlert("Ошибка", "Заполните все
обязательные поля", "OK");
            return;
        }

        var dto = new RegisterRequestDto
        {
            Email = email,
            Password = password,
            LastName = lastName,
            FirstName = firstName,
            MiddleName = middleName,
            PhoneNumber = phone
        };

        try
        {
            using var client = new HttpClient();
            var json = JsonSerializer.Serialize(dto);
            var content = new StringContent(json, Encoding.UTF8,
"application/json");

            var response = await
client.PostAsync($"{AppSettings.ApiBaseUrl}/Auth/register",
content);

            if (response.IsSuccessStatusCode)
            {
                await DisplayAlert("Успешно", "Регистрация прошла
успешно", "OK");
                await Navigation.PopAsync(); // возвращение к
странице входа
            }
            else
            {
                var error = await
response.Content.ReadAsStringAsync();
                await DisplayAlert("Ошибка", error, "OK");
            }
        }
        catch (Exception ex)
        {
            await DisplayAlert("Ошибка", $"Ошибка подключения:
{ex.Message}", "OK");
        }
    }
}

```

```

private async void OnLoginClicked(object sender, EventArgs e)
{
    await Navigation.PopAsync();
}
}

```

Листинг Г.22 – Файл ServicesPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        x:Class="VetClinicAppMobile.Pages.ServicesPage"
        Title="Список услуг">
    <RefreshView IsRefreshing="{Binding IsRefreshing}"
        Command="{Binding LoadServicesCommand}"
        Margin="10">
        <CollectionView x:Name="ServicesCollection"
            ItemsSource="{Binding Services}"
            SelectionMode="Single"

SelectionChanged="ServicesCollection_SelectionChanged">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame CornerRadius="15"
                        Padding="15"
                        Margin="0,8"
                        BackgroundColor="White"
                        HasShadow="True">
                        <Grid RowDefinitions="Auto,Auto,Auto"
                            ColumnDefinitions="*,Auto"
                            RowSpacing="5">
                            <!-- Название услуги -->
                            <Label Text="{Binding ServicName}"
                                FontSize="18"
                                FontAttributes="Bold"
                                TextColor="#2C3E50"
                                Grid.Row="0"
                                Grid.Column="0"/>
                            <!-- Цена -->
                            <Label Text="{Binding Price, StringFormat='{0} ₸'}"
                                FontSize="18"
                                FontAttributes="Bold"
                                TextColor="#191970"
                                Grid.Row="0"
                                Grid.Column="1"
                                HorizontalOptions="End"/>
                            <!-- Разделитель -->
                            <BoxView HeightRequest="1"
                                BackgroundColor="#ECF0F1"
                                Grid.Row="1"
                                Grid.ColumnSpan="2"/>
                            <!-- Описание -->

```

```

        <Label Text="{Binding ServiceDescription}"
                FontSize="14"
                TextColor="#7F8C8D"
                Grid.Row="2"
                Grid.ColumnSpan="2"
                MaxLines="3"
                LineBreakMode="TailTruncation"/>
    </Grid>
</Frame>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</RefreshView>
</ContentPage>

```

Листинг Г.23 – Файл ServicesPage.xaml.cs

```

using System.Collections.ObjectModel;
using VetClinicAppMobile.Services;

namespace VetClinicAppMobile.Pages;

public partial class ServicesPage : ContentPage
{
    private readonly HttpClient _httpClient;
    public ObservableCollection<VetServiceDto> Services { get; } =
new();

    public ServicesPage(HttpClient httpClient)
    {
        InitializeComponent();
        _httpClient = httpClient;
        BindingContext = this;
        LoadServicesCommand = new Command(async () => await
LoadServicesAsync());
        LoadServicesCommand.Execute(null);
    }

    public LoadServicesCommand { get; }

    private async Task LoadServicesAsync()
    {
        var response = await
_httpClient.GetFromJsonAsync<List<VetServiceDto>>("api/VetServices
");
        Services.Clear();
        foreach (var service in response ?? new
List<VetServiceDto>())
            Services.Add(service);
    }
    private async void ServicesCollection_SelectionChanged(object
sender, SelectionChangedEventArgs e)
    {

```



```

        BorderColor="#2196F3"
        CornerRadius="12"
        Padding="12"
        HasShadow="True">
<VerticalStackLayout>
    <Label Text="{Binding Date,
StringFormat='{0:dd.MM.yyyy}'}"
        FontAttributes="Bold"
        FontSize="16"
        TextColor="#0d47a1" />
    <Label Text="{Binding
Description}" FontSize="14" TextColor="#0d47a1" />
</VerticalStackLayout>
</Frame>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>
</VerticalStackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.25 – Файл VisitHistoryPage.xaml.cs

```

using System.Collections.ObjectModel;
using VetClinicAppMobile.Models;

namespace VetClinicAppMobile.Pages;

public partial class VisitHistoryPage : ContentPage
{
    private readonly HttpClient _httpClient = new();
    public event PropertyChangedEventHandler PropertyChanged;

    public ObservableCollection<VisitDto> Visits { get; set; } =
new();
    public PetDto Pet { get; set; } = new();

    public VisitHistoryPage()
    {
        InitializeComponent();
        BindingContext = this;
        LoadData();
    }

    private async void LoadData()
    {
        try
        {
            var petResponse = await
_httpClient.GetFromJsonAsync<PetDto>($"/api/pets/{PetId}");
            var visitsResponse = await
_httpClient.GetFromJsonAsync<List<VisitDto>>($"/api/visits/pet/{Pe
tId}");
            if (petResponse != null)

```

```

        Pet = petResponse;
        Visits.Clear();
        foreach (var visit in visitsResponse ?? new
List<VisitDto>())
        {
            Visits.Add(visit);
            OnPropertyChanged(nameof(Pet));
            OnPropertyChanged(nameof(Visits));
        }
        catch (Exception ex)
        {
            await DisplayAlert("Ошибка", "Не удалось загрузить
данные", "OK");
        }
    }
    protected virtual void OnPropertyChanged(string propertyName)
=>
    {
        PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
    }
}

```

Листинг Г.26 – Файл AddDoctorPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="VetClinicAppMobile.Pages.AddDoctorPage"
Title="Ветеринарные врачи">
    <ScrollView>
        <VerticalStackLayout Padding="20" Spacing="16">
            <Label Text="Выбор врача:" FontSize="18"
FontAttributes="Bold" TextColor="#2a2a2a"/>
            <Entry Placeholder="Введите или выберите врача"
BackgroundColor="#e6f0ff" />
            <Button Text="Удалить врача"
BackgroundColor="#ffcccc"
TextColor="Black"
CornerRadius="8"/>
            <Label Text="Добавление врача:" FontSize="18"
FontAttributes="Bold" TextColor="#2a2a2a"/>
            <Entry x:Name="LastNameEntry" Placeholder="Фамилия"
BackgroundColor="#e6f0ff"/>
            <Entry x:Name="FirstNameEntry" Placeholder="Имя"
BackgroundColor="#e6f0ff"/>
            <Entry x:Name="MiddleNameEntry" Placeholder="Отчество"
BackgroundColor="#e6f0ff"/>
            <Entry x:Name="PhoneEntry" Placeholder="Номер
телефона" Keyboard="Telephone" BackgroundColor="#e6f0ff"/>
            <Entry x:Name="EmailEntry" Placeholder="Email"
Keyboard="Email" BackgroundColor="#e6f0ff"/>
        </VerticalStackLayout>
    </ScrollView>
</ContentPage>

```

```

        <Entry x:Name="SpecializationEntry"
Placeholder="Специализация" BackgroundColor="#e6f0ff"/>
        <Entry x:Name="ServicesEntry" Placeholder="Услуги"
BackgroundColor="#e6f0ff"/>
        <Button Text="Добавить врача"
                BackgroundColor="#ccffcc"
                TextColor="Black"
                CornerRadius="8"
                Clicked="OnAddDoctorClicked"/>

    </VerticalStackLayout>
</ScrollView>
</ContentPage>

```

Листинг Г.27 – Файл AddDoctorPage.xaml.cs

```

using VetClinicAppMobile.Models;
using VetClinicAppMobile.Services;

namespace VetClinicAppMobile.Pages;

public partial class AddDoctorPage : ContentPage
{
    private readonly ApiService _apiService = new();

    public AddDoctorPage()
    {
        InitializeComponent();
    }

    private async void OnAddDoctorClicked(object sender, EventArgs
e)
    {
        var doctor = new AddDoctorDto
        {
            FirstName = FirstNameEntry.Text,
            LastName = LastNameEntry.Text,
            MiddleName = MiddleNameEntry.Text,
            PhoneNumber = PhoneEntry.Text,
            Email = EmailEntry.Text,
            Specialization = SpecializationEntry.Text,
            Services = ServicesEntry.Text
        };

        var response = await _apiService.PostAsync("Doctors",
doctor);

        if (response.IsSuccessStatusCode)
        {
            await DisplayAlert("Успех", "Врач добавлен", "OK");
            ClearFields();
        }
        else
        {

```

```

        await DisplayAlert("Ошибка", "Не удалось добавить
врача", "OK");
    }
}
private void ClearFields()
{
    FirstNameEntry.Text = "";
    LastNameEntry.Text = "";
    MiddleNameEntry.Text = "";
    PhoneEntry.Text = "";
    EmailEntry.Text = "";
    SpecializationEntry.Text = "";
    ServicesEntry.Text = "";
}
}

```

Листинг Г.28 – Файл AddServicePage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
            x:Class="VetClinicAppMobile.Pages.AddServicePage"
            Title="Услуги">
    <ScrollView>
        <VerticalStackLayout Padding="20" Spacing="16">
            <Label Text="Выбор услуги:" FontSize="18"
FontAttributes="Bold" TextColor="#2a2a2a"/>
            <Entry Placeholder="Введите или выберите услугу"
BackgroundColor="#e6f0ff" />
            <Button Text="Удалить услугу"
                    BackgroundColor="#ffcccc"
                    TextColor="Black"
                    CornerRadius="8"/>
            <Label Text="Добавление услуги:" FontSize="18"
FontAttributes="Bold" TextColor="#2a2a2a"/>

            <Entry x:Name="ServiceNameEntry"
Placeholder="Название" BackgroundColor="#e6f0ff"/>
            <Entry x:Name="ServicePriceEntry" Placeholder="Цена"
Keyboard="Numeric" BackgroundColor="#e6f0ff"/>
            <Editor x:Name="ServiceDescriptionEditor"
Placeholder="Описание" HeightRequest="100"
BackgroundColor="#e6f0ff"/>
            <Button Text="Добавить услугу"
                    BackgroundColor="#ccffcc"
                    TextColor="Black"
                    CornerRadius="8"
                    Clicked="OnAddServiceClicked"/>
        </VerticalStackLayout>
    </ScrollView>
</ContentPage>

```

Листинг Г.29 – Файл AddServicePage.xaml.cs

```

using VetClinicAppMobile.Models;
using VetClinicAppMobile.Services;

namespace VetClinicAppMobile.Pages;

public partial class AddServicePage : ContentPage
{
    private readonly ApiService _apiService = new();

    public AddServicePage()
    {
        InitializeComponent();
    }

    private async void OnAddServiceClicked(object sender,
EventArgs e)
    {
        var service = new AddServiceDto
        {
            Name = ServiceNameEntry.Text,
            Price = double.TryParse(ServicePriceEntry.Text, out
double p) ? p : 0,
            Description = ServiceDescriptionEditor.Text
        };
        var response = await _apiService.PostAsync("Services",
service);

        if (response.IsSuccessStatusCode)
        {
            await DisplayAlert("Успех", "Услуга добавлена", "OK");
            ClearFields();
        }
        else
        {
            await DisplayAlert("Ошибка", "Не удалось добавить
услугу", "OK");
        }
    }

    private void ClearFields()
    {
        ServiceNameEntry.Text = "";
        ServicePriceEntry.Text = "";
        ServiceDescriptionEditor.Text = "";
    }
}

```

Листинг Г.30 – Файл AppShell.xaml

```

<?xml version="1.0" encoding="UTF-8" ?>
<Shell
    x:Class="VetClinicAppMobile.AppShell"
    xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:local="clr-namespace:VetClinicAppMobile"
    xmlns:pages="clr-namespace:VetClinicAppMobile.Pages"

```

```

Shell.FlyoutBehavior="Disabled"
Title="VetClinicAppMobile">

    <ShellContent Title="Login" ContentTemplate="{DataTemplate
pages:LoginPage}" Route="LoginPage" />

    <!-- Панель вкладок только для клиента -->
    <TabBar x:Name="ClientTabBar">
        <ShellContent Title="Приемы" Icon="appointments.png"
ContentTemplate="{DataTemplate pages:AppointmentsPage}"
Route="AppointmentsPage"/>
        <ShellContent Title="Запись" Icon="booking.png"
ContentTemplate="{DataTemplate pages:BookingPage}"
Route="BookingPage"/>
        <ShellContent Title="Услуги" Icon="services.png"
ContentTemplate="{DataTemplate pages:ServicesPage}"
Route="ServicesPage"/>
        <ShellContent Title="Профиль" Icon="profile.png"
ContentTemplate="{DataTemplate pages:ProfilePage}"
Route="ProfilePage"/>
    </TabBar>

    <!-- Панель вкладок для доктора -->
    <TabBar x:Name="DoctorTabBar">
        <ShellContent Title="Приемы" Icon="appointments.png"
ContentTemplate="{DataTemplate pages:AddPetVisitPage}"
Route="AddPetVisitPage"/>
        <ShellContent Title="Расписание" Icon="booking.png"
ContentTemplate="{DataTemplate pages:DoctorSchedulePage}"
Route="DoctorSchedulePage"/>
        <ShellContent Title="Пациенты" Icon="services.png"
ContentTemplate="{DataTemplate pages:VisitHistoryPage}"
Route="VisitHistoryPage"/>
        <ShellContent Title="Профиль" Icon="profile.png"
ContentTemplate="{DataTemplate pages:ProfilePage}"
Route="ProfilePage"/>
    </TabBar>

    <!-- Панель вкладок для администратора -->
    <TabBar x:Name="AdminTabBar">
        <ShellContent Title="Приемы" Icon="appointments.png"
ContentTemplate="{DataTemplate pages:AddPetVisitPage}"
Route="AddPetVisitPage"/>
        <ShellContent Title="Расписание" Icon="booking.png"
ContentTemplate="{DataTemplate pages:DoctorSchedulePage}"
Route="DoctorSchedulePage"/>
        <ShellContent Title="Пациенты" Icon="services.png"
ContentTemplate="{DataTemplate pages:VisitHistoryPage}"
Route="VisitHistoryPage"/>
        <ShellContent Title="Профиль" Icon="profile.png"
ContentTemplate="{DataTemplate pages:AddServicePage}"
Route="AddServicePage"/>
    </TabBar>

```

```

        <ShellContent Title="Admin" ContentTemplate="{DataTemplate
pages:AdminDashboard}" Route="AdminDashboard" />
        <ShellContent ContentTemplate="{DataTemplate
pages:AppointmentDetailPage}" Route="AppointmentDetailPage"/>
        <ShellContent ContentTemplate="{DataTemplate
pages:VisitHistoryPage}" Route="VisitHistoryPage"/>

</Shell>

```

Листинг Г.31 – Файл AppShell.xaml.cs

```

namespace VetClinicAppMobile
{
    public partial class AppShell : Shell
    {
        public AppShell()
        {
            InitializeComponent();
            SetClientTabsVisible(false);
            SetDoctorTabsVisible(false);
            SetAdminTabsVisible(false);
        }
        public void SetClientTabsVisible(bool isVisible)
        {
            ClientTabBar.IsVisible = isVisible;
        }

        public void SetDoctorTabsVisible(bool isVisible)
        {
            DoctorTabBar.IsVisible = isVisible;
        }
        public void SetAdminTabsVisible(bool isVisible)
        {
            DoctorTabBar.IsVisible = isVisible;
        }
    }
}

```

Приложение Д
(рекомендуемое)

Результаты тестирования серверной части приложения

```
Curl curl -X 'POST' \
      'http://localhost:5257/api/Auth/login' \
      -H 'accept: */*' \
      -H 'Content-Type: application/json' \
      -d '{
        "email": "terapevt@gmail.com",
        "password": "paroll23"
      }'
```

Request URL

http://localhost:5257/api/Auth/login

Server response

Code Details

200

Response body

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IWRlcXVCJ9.eyJodWQiOiBvc2NoZWlhcys5MmVrbzZfLW9yZy93cy9kaDI1Ail12kZm50aXR5L2NsYm1tcy9uYm1lIjoidGVyYXB1dnRAZ21haWwvY29tIiwiaWF0IjoiMTYxOTQ1MzRXRDh6G1lbnQ1Q.6vC7NKxoUhd-8GCKdkKamWiso5w7cVatGQQOpymHook",
  "role": "Doctor",
  "userId": 14,
  "clientId": null,
  "doctorId": "",
  "adminId": null
}
```

Response headers

```
access-control-allow-origin: *
content-type: application/json; charset=utf-8
date: Sat, 31 May 2025 17:43:35 GMT
server: Kestrel
transfer-encoding: chunked
```

Рисунок Д.2 – Авторизация ветеринарного врача

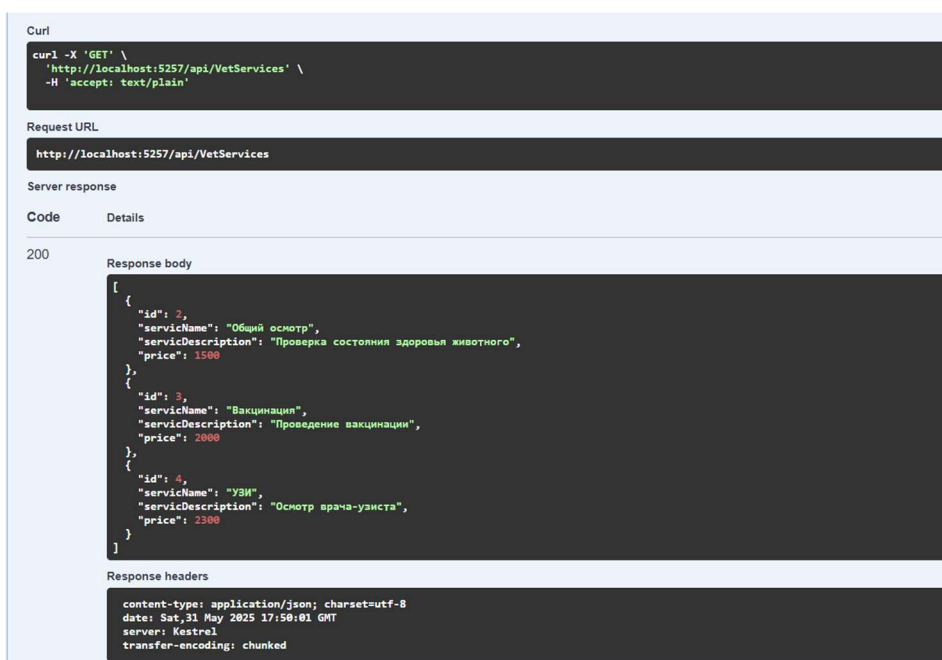


Рисунок Д.3 – Получение списка услуг

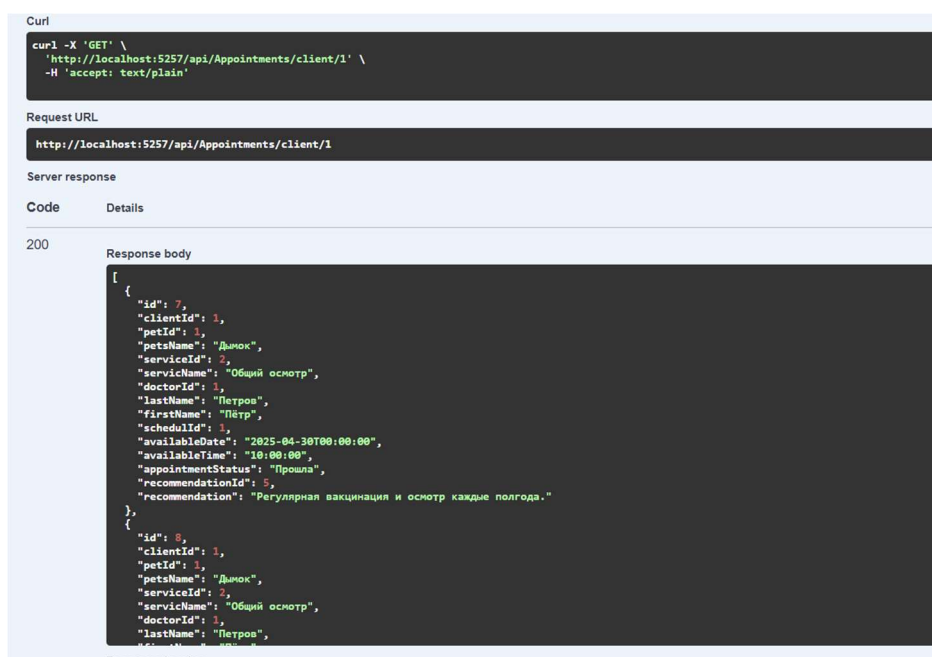


Рисунок Д.4 – Получение списка записей

Curl

```
curl -X 'GET' \
  'http://localhost:5257/api/DoctorSchedules/doctor/1/date/2025-05-21T00:3A00X3A00' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:5257/api/DoctorSchedules/doctor/1/date/2025-05-21T00:3A00X3A00
```

Server response

Code	Details
200	<p>Response body</p> <pre>[{ "id": 7, "doctorId": 1, "availableDate": "2025-05-21T00:00:00", "availableTime": "17:00:00", "isBooked": true, "doctor": null }]</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Sat, 31 May 2025 17:54:00 GMT server: Kestrel transfer-encoding: chunked</pre>

Рисунок Д.5 – Получение доступного времени для записи

Curl

```
curl -X 'GET' \
  'http://localhost:5257/api/Client/1' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5257/api/Client/1
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": 1, "userId": 1, "lastName": "Иванов", "firstName": "Иван", "middleName": "Иванович", "phoneNumber": "+79991112233", "email": "client@example.com" }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Sat, 31 May 2025 17:56:16 GMT server: Kestrel transfer-encoding: chunked</pre>

Рисунок Д.6 – Получение информации о клиенте

```
curl -X 'POST' \
  'http://localhost:5257/api/Auth/register' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "email": "clienttest",
    "password": "pass_234",
    "lastName": "Евон",
    "firstName": "Лев",
    "middleName": "Иванович",
    "phoneNumber": "86756893345"
  }'
```

Request URL

```
http://localhost:5257/api/Auth/register
```

Server response

Code	Details
200	<p>Response body</p> <pre>Регистрация прошла успешно</pre> <p>Response headers</p> <pre>access-control-allow-origin: * content-type: text/plain; charset=utf-8 date: Sat, 31 May 2025 19:03:00 GMT server: Kestrel transfer-encoding: chunked</pre>

Рисунок Д.7 – Регистрация нового клиента

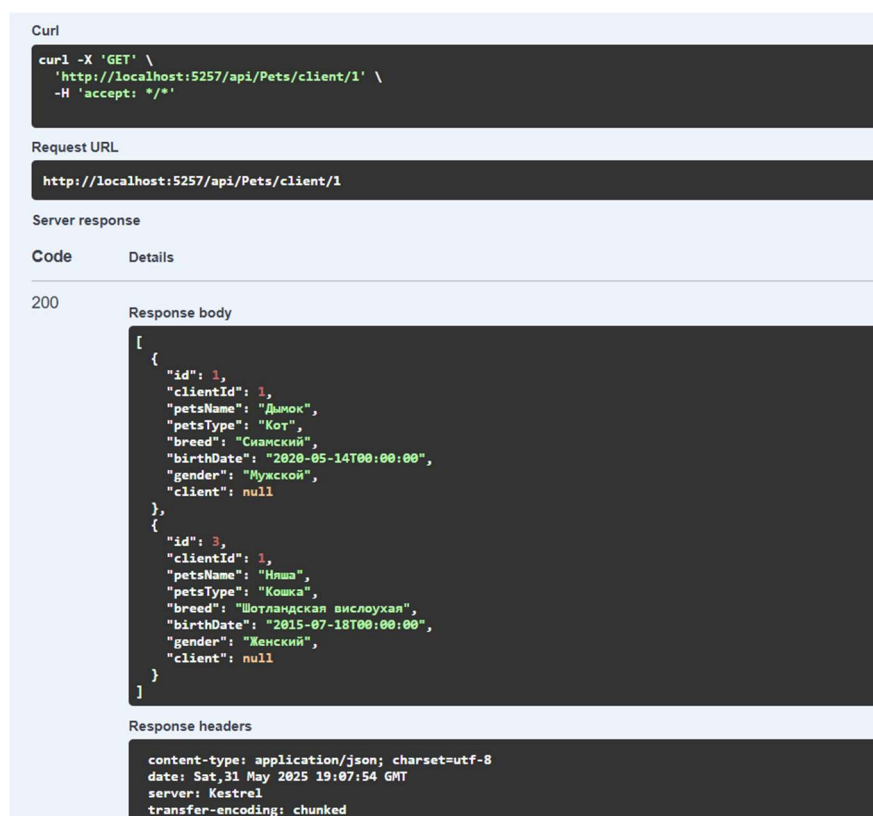


Рисунок Д.8 – Получение списка питомцев клиента

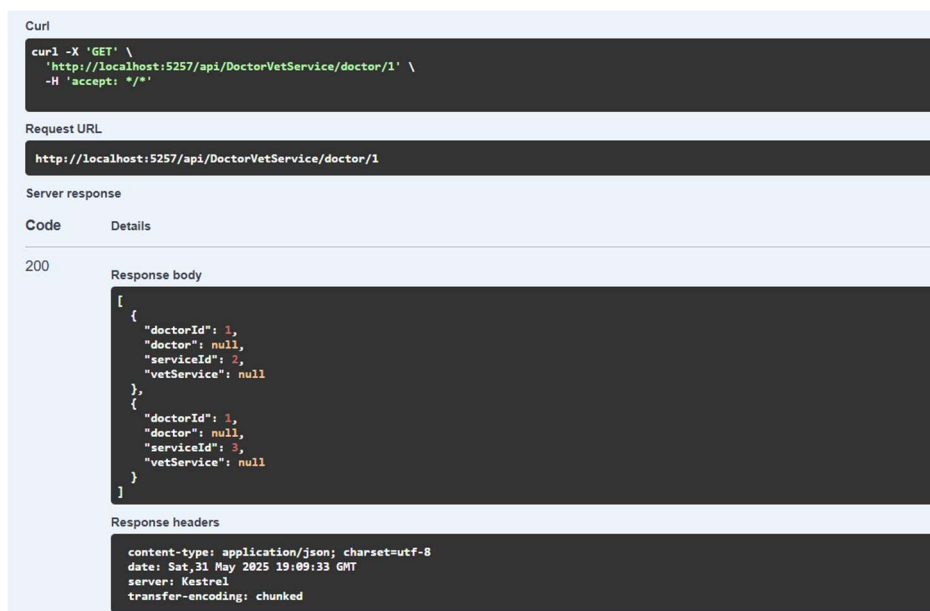


Рисунок Д.9 – Получение списка услуг, предоставляемых врачом

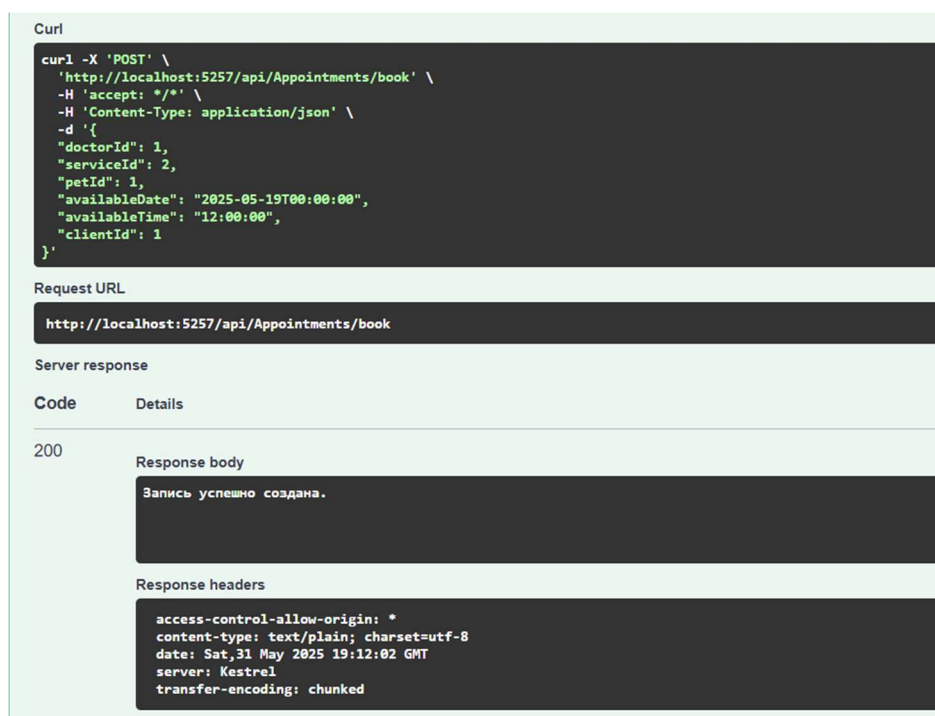


Рисунок Д.10 – Добавление записи к врачу

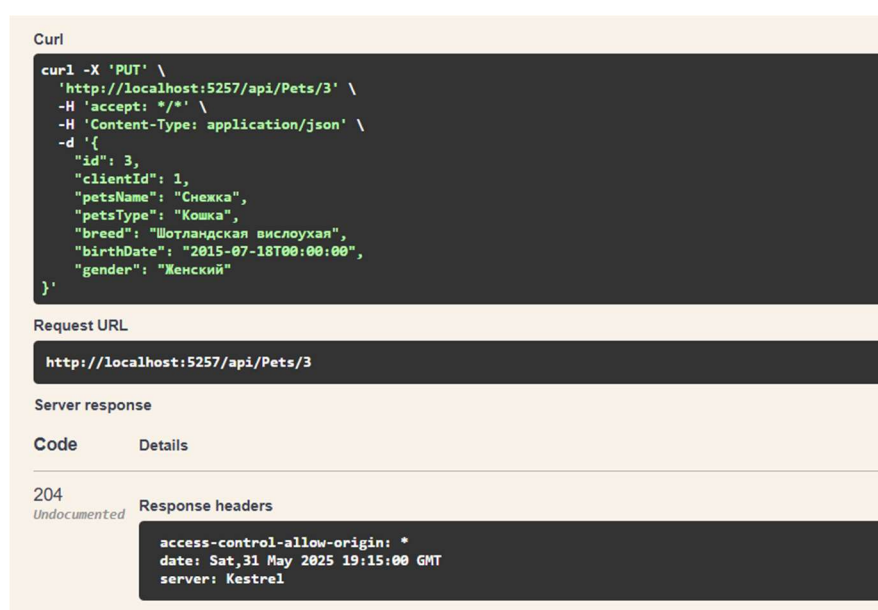


Рисунок Д.11 – Редактирование существующего питомца

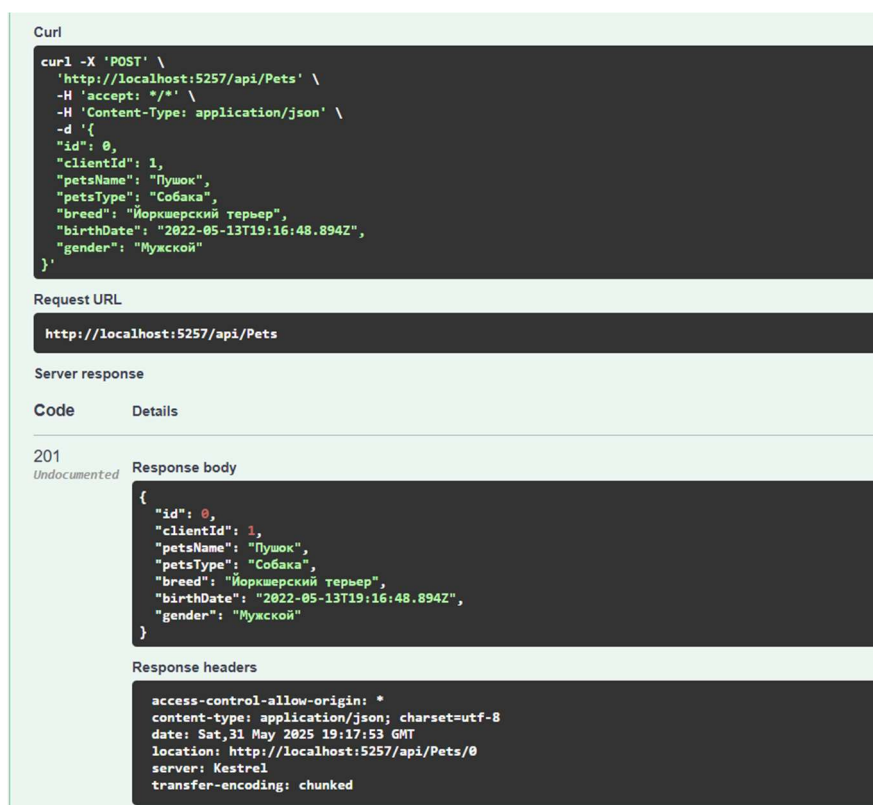


Рисунок Д.12 – Добавление нового питомца

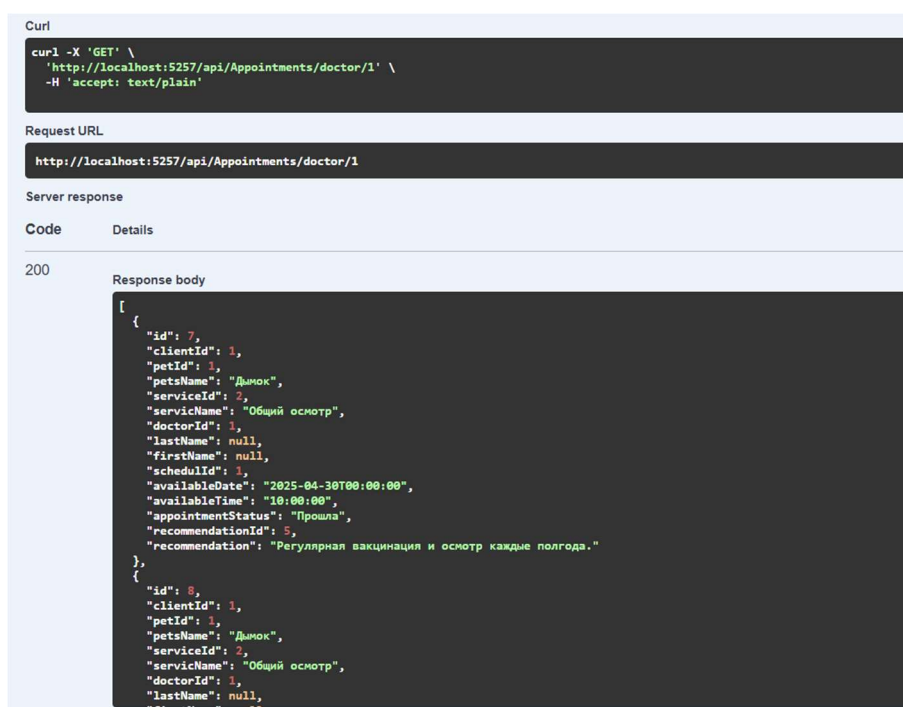


Рисунок Д.13 – Получение списка записей ветеринарного врача

Curl

```
curl -X 'POST' \
  'http://localhost:5257/api/DoctorSchedules' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "doctorId": 1,
    "availableDate": "2025-06-05T19:21:28.154Z",
    "availableTime": "12:00:00",
    "isBooked": false
  }'
```

Request URL

`http://localhost:5257/api/DoctorSchedules`

Server response

Code	Details
201 <i>Undocumented</i>	<p>Response body</p> <pre>{ "id": 0, "doctorId": 1, "availableDate": "2025-06-05T19:21:28.154Z", "availableTime": "12:00:00", "isBooked": false }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * content-type: application/json; charset=utf-8 date: Sat,31 May 2025 19:21:59 GMT location: http://localhost:5257/api/DoctorSchedules/0 server: Kestrel transfer-encoding: chunked</pre>

Рисунок Д.14 – Добавление нового слота для записи к врачу

Curl

```
curl -X 'POST' \
  'http://localhost:5257/api/Doctors' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 0,
    "userId": 17,
    "lastName": "Львов",
    "firstName": "Кирилл",
    "middleName": "Петрович",
    "specialization": "Врач-кардиолог",
    "phoneNumber": "895678564826",
    "email": "lvovkir@mail.com"
  }'
```

Request URL

`http://localhost:5257/api/Doctors`

Server response

Code	Details
201 <i>Undocumented</i>	<p>Response body</p> <pre>{ "id": 0, "userId": 17, "lastName": "Львов", "firstName": "Кирилл", "middleName": "Петрович", "specialization": "Врач-кардиолог", "phoneNumber": "895678564826", "email": "lvovkir@mail.com" }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * content-type: application/json; charset=utf-8 date: Sat,31 May 2025 19:27:10 GMT location: http://localhost:5257/api/Doctors/0 server: Kestrel transfer-encoding: chunked</pre>

Рисунок Д.15 – Добавление нового врача

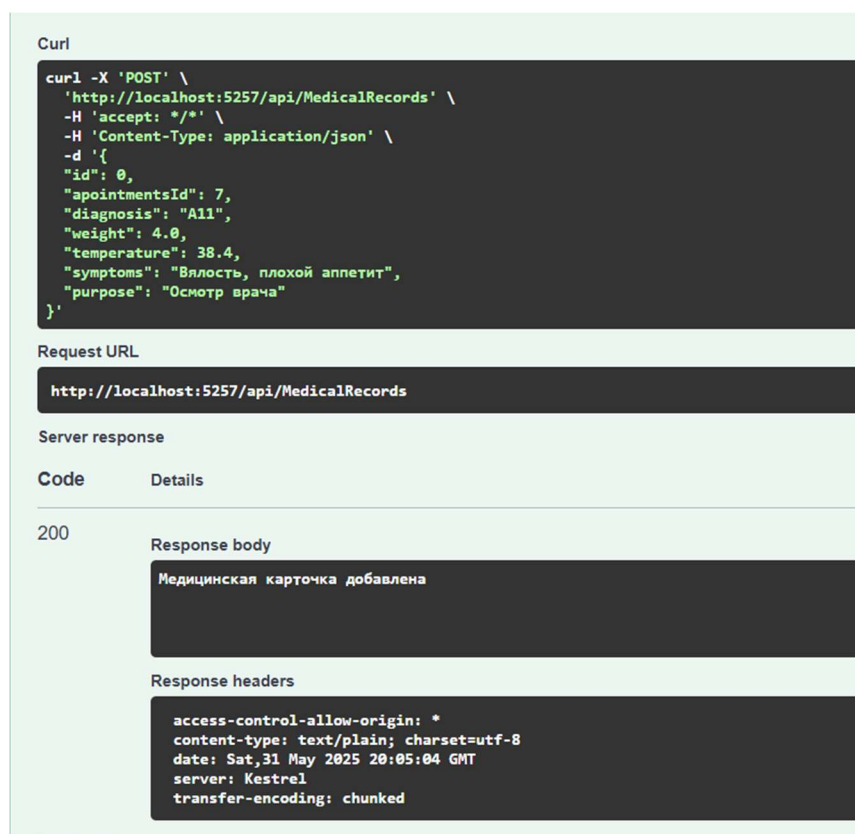


Рисунок Д.16 – Добавление новой медицинской карточки

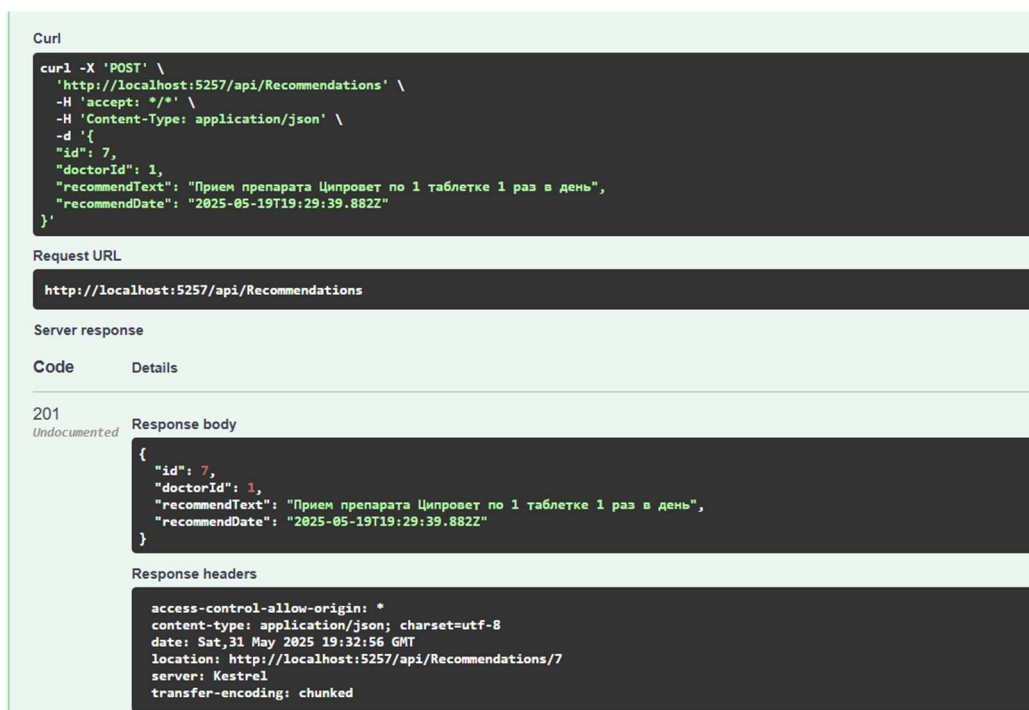


Рисунок Д.17 – Добавление новой рекомендации

Curl

```
curl -X 'DELETE' \
'http://localhost:5257/api/Doctors/9' \
-H 'accept: */*'
```

Request URL

```
http://localhost:5257/api/Doctors/9
```

Server response

Code	Details
204 <i>Undocumented</i>	<p>Response headers</p> <pre>access-control-allow-origin: * date: Sat, 31 May 2025 19:34:43 GMT server: Kestrel</pre>

Responses

Code	Description
200	OK

Рисунок Д.18 – Удаление врача

Приложение Е
(рекомендуемое)
Исходный код функций тестирования

```

using NUnit.Framework;
using VetClinicApp;

namespace VetClinicTest
{
    [TestFixture]
    public class VetClinicTests
    {
        private FakeAppointmentService _service;

        [SetUp]
        public void Setup()
        {
            _service = new FakeAppointmentService();
        }

        [Test]
        public void
GetAppointmentsByClientId_ShouldReturnAppointments()
        {
            var result = _service.GetAppointmentsByClientId(1);
            Assert.That(result, Is.Not.Empty);
        }

        [Test]
        public void
GetAppointmentsByClientId_ShouldReturnEmpty_WhenNotFound()
        {
            var result = _service.GetAppointmentsByClientId(999);
            Assert.That(result, Is.Empty);
        }

        [Test]
        public void GetAllServices_ShouldReturnServices()
        {
            var services = _service.GetAllServices();
            Assert.That(services.Count(), Is.EqualTo(2));
        }

        [Test]
        public void GetPetsByClientId_ShouldReturnCorrectPets()
        {
            var pets = _service.GetPetsByClientId(1);
            Assert.That(pets.Count(), Is.EqualTo(2));
        }

        [Test]
        public void BookAppointment_ShouldSucceed_IfSlotIsFree()
        {
            var result = _service.BookAppointment(new
BookAppointmentDto
            {
                ClientId = 1,

```

```

        PetId = 2,
        ServiceId = 2,
        DoctorId = 101,
        Date = DateTime.Today,
        Time = new TimeSpan(11, 0, 0)
    ));

    Assert.That(result, Is.True);
}

[Test]
public void BookAppointment_ShouldFail_IfSlotIsTaken()
{
    var result = _service.BookAppointment(new
BookAppointmentDto
    {
        ClientId = 1,
        PetId = 1,
        ServiceId = 1,
        DoctorId = 101,
        Date = DateTime.Today,
        Time = new TimeSpan(10, 0, 0)
    });

    Assert.That(result, Is.False);
}

[Test]
public void
BookAppointment_ShouldIncreaseAppointmentCount()
{
    var before =
_service.GetAppointmentsByClientId(1).Count();

    _service.BookAppointment(new BookAppointmentDto
    {
        ClientId = 1,
        PetId = 2,
        ServiceId = 2,
        DoctorId = 101,
        Date = DateTime.Today,
        Time = new TimeSpan(12, 0, 0)
    });

    var after =
_service.GetAppointmentsByClientId(1).Count();
    Assert.That(after, Is.EqualTo(before + 1));
}
}
}

```

Приложение Ж
(рекомендуемое)

Результаты тестирования мобильного приложения

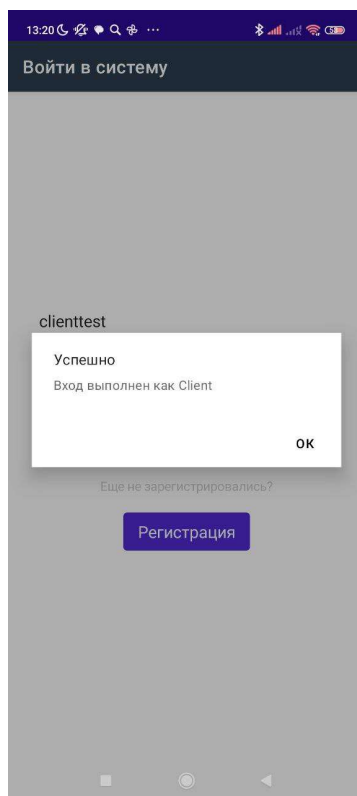


Рисунок Ж.1 – Авторизация клиента с корректными данными

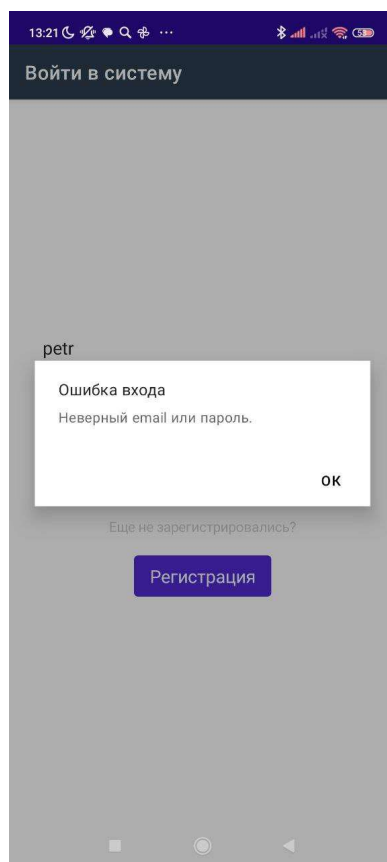


Рисунок Ж.2 – Авторизация клиента с некорректными данными

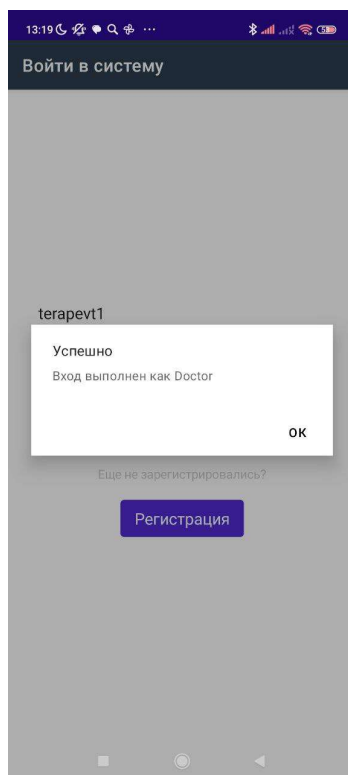


Рисунок Ж.3 – Авторизация врача с корректными данными

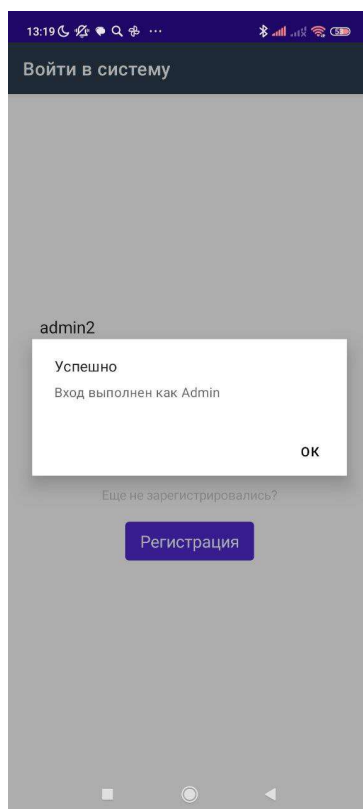


Рисунок Ж.4 – Авторизация администратора с корректными данными

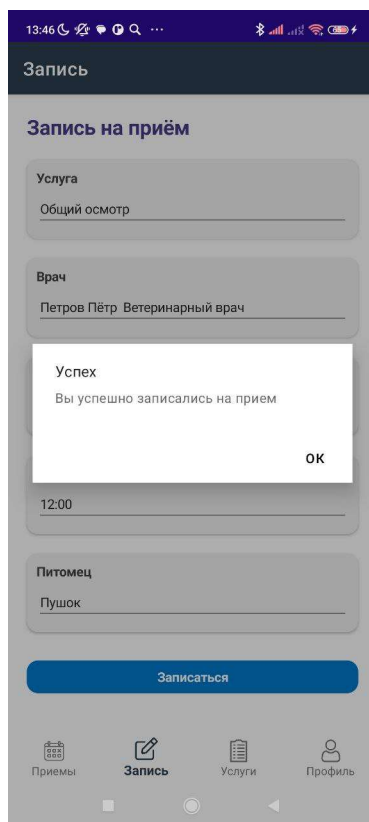


Рисунок Ж.5 – Запись к ветеринарному врачу

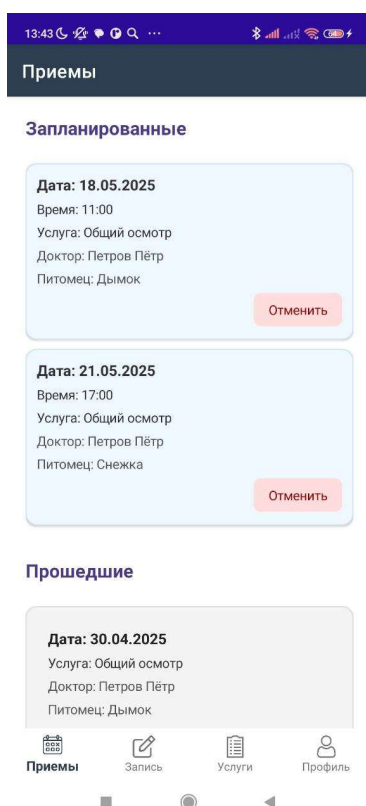


Рисунок Ж.6 – Просмотр истории посещений



Рисунок Ж.7 – Просмотр услуг и цен на них

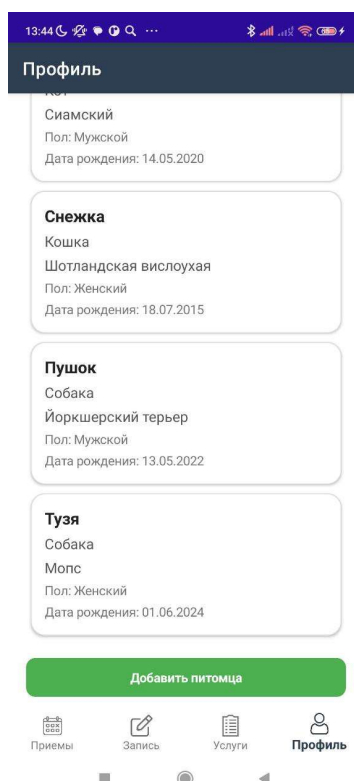


Рисунок Ж.8 – Добавление питомца

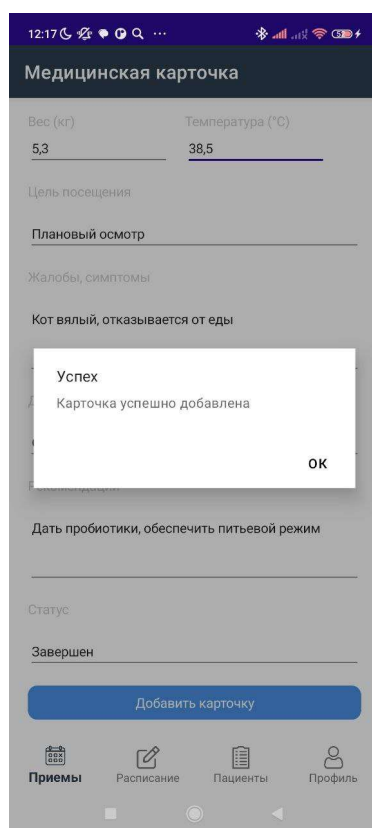


Рисунок Ж.9 – Добавление медицинской карточки питомца

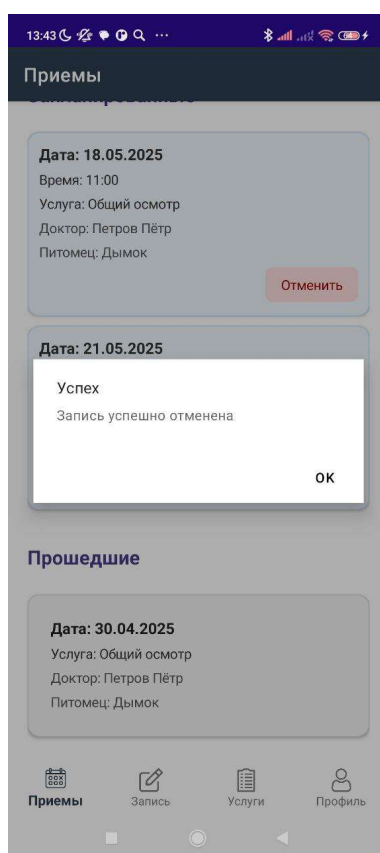


Рисунок Ж.10 – Отмена записи к ветеринарному врачу

12:15

Моё расписание

☐ Четверг

☒ Пятница

С:

09:00

До:

14:00

Длительность слота (мин):

30

Добавить расписание

Добавленные слоты:

Понедельник, 03.06.2025
09:00 – 09:30

Понедельник, 03.06.2025
09:30 – 10:00

Приемы Расписание Пациенты Профиль

Рисунок Ж.11 – Добавление расписания

13:43

← Прием

30.04.2025

Услуга:
Общий осмотр

Врач:
Петров Пётр

Питомец:
Дымок

Рекомендации врача
Регулярная вакцинация и осмотр каждые полгода.

Приемы Запись Услуги Профиль

Рисунок Ж.12 – Просмотр рекомендаций врача

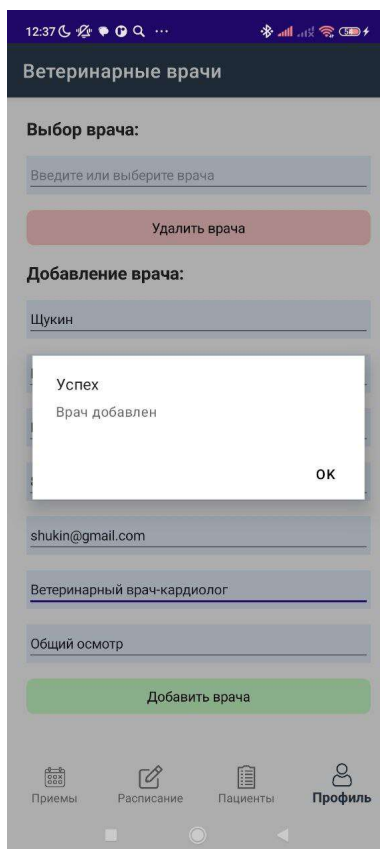


Рисунок Ж.13 – Добавление ветеринарного врача

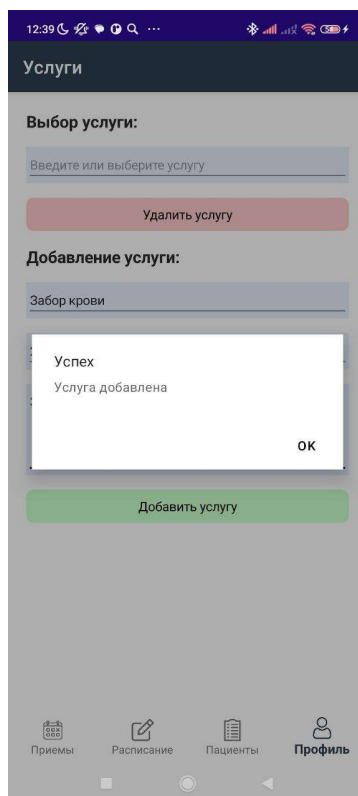


Рисунок Ж.14 – Добавление услуги